

24.11.2004

日 本 国 特 許 庁
JAPAN PATENT OFFICE

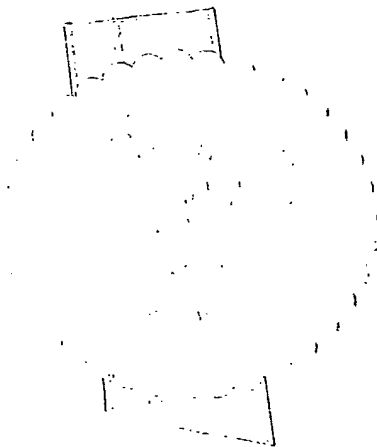
別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 1 1 月 1 9 日
Date of Application:

出 願 番 号 特 願 2 0 0 3 - 3 8 9 2 6 4
Application Number:
[ST. 10/C] : [J P 2 0 0 3 - 3 8 9 2 6 4]

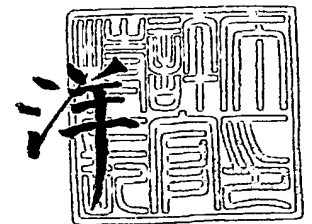
出 願 人 財団法人北九州産業学術推進機構
Applicant(s):



2 0 0 5 年 1 月 1 3 日

特許庁長官
Commissioner,
Japan Patent Office

小 川



BEST AVAILABLE COPY

【書類名】 特許願
【整理番号】 JP030038
【あて先】 特許庁長官殿
【国際特許分類】 G06F 15/60
G01R 31/28

【発明者】
【住所又は居所】 福岡県福岡市中央区鳥飼 3 丁目 1 5 - 1 7
【氏名】 笹尾 勤

【発明者】
【住所又は居所】 神奈川県足柄下郡箱根町強羅 1 3 2 0 - 1 2 3 1
【氏名】 井口 幸洋

【特許出願人】
【識別番号】 802000031
【氏名又は名称】 財団法人北九州産業学術推進機構

【代理人】
【識別番号】 100121371
【弁理士】
【氏名又は名称】 石田 和人
【電話番号】 093-695-3113

【手数料の表示】
【予納台帳番号】 222495
【納付金額】 21,000円

【提出物件の目録】
【物件名】 特許請求の範囲 1
【物件名】 明細書 1
【物件名】 図面 1
【物件名】 要約書 1

【書類名】特許請求の範囲

【請求項 1】

入力変数を $X=(x_1, \dots, x_n)$ ($n \in \text{自然数}$) とする多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して (数 1) により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2, m \in \text{自然数}$) は $F(X)$ の出力を表す変数。) を表現する特性関数二分決定グラフ (Binary Decision Diagram for Characteristic Function: BDD for CF) を格納する記憶手段であって、前記特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$) に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$) の値が 0 のとき及び 1 のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ の組からなる節点データが記憶されたものである節点テーブル記憶手段と、

前記多出力論理関数 $F(X)$ に対応する論理回路を構成するためのデータであるルックアップ・テーブル (Look-Up Table: LUT) を格納するための LUT 記憶手段と、

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを所定の高さ lev の分割線において 2 つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものであって、出力を表す変数 y_r ($\in Y$) に関わる節点 v_j 及びその親節点 v_k の節点データについて、節点 v_j の枝 $e_0(v_j)$, $e_1(v_j)$ の一方 $e_a(v_j)$ が $\chi(X, Y)=0$ に関わる終端節点を特定する場合、当該節点 v_j の親節点 v_k の枝 $e_0(v_k)$, $e_1(v_k)$ のうち当該節点を特定する枝 $e_c(v_k)$ を、当該節点 v_j の枝 $e_a(v_j)$ 以外の枝 $e_b(v_j)$ に置き換える短絡除去処理を行う短絡除去手段と、

前記短絡除去手段により短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって高さ lev より高い高さにある非終端節点に属する枝のうち、高さ lev より低い高さの非終端節点の子節点を特定するものの個数を計数し (但し、同じ節点を特定するものは 1 つと数える。)、その個数を高さ lev の分割線における幅 W として出力する BDD 幅計測手段と、

前記 BDD 幅計測手段が出力する幅 W に基づき、(数 2) の演算により中間変数の個数 u を算出する中間変数算出手段と、

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを所定の高さ lev の分割線において 2 つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものについて、ルックアップ・テーブルを生成しそれを LUT 記憶手段に格納する LUT 生成手段と、

前記中間変数算出手段が算出する前記中間変数の個数 u と等しい制御入力数を有する二分木 (binary tree) を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフ B_0 に属する非終端節点の節点データを、前記二分木を表す節点データで置き換えることにより、特性関数二分決定グラフを再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新する BDD 再構成手段と、
を備えていることを特徴とする論理回路合成装置。

【数 1】

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} (y_i \equiv f_i(X))$$

但し、 $Y=(y_0, \dots, y_{m-1})$, $F=(f_0(X), \dots, f_{m-1}(X))$, $m \in \text{整数}$

【数 2】

$$u = \lceil \log_2 W \rceil$$

【請求項 2】

前記多出力論理関数 $F(X)$ の要素をなす論理関数 $f_0(X), \dots, f_{m-1}(X)$ の順序を $\pi = (\pi[0], \dots, \pi[m-1])$ (但し、 $\pi[i]=j$ は、 f_j が i 番目であることを表す。) とし、論理関数 f_j ($\in F(X)$) が依存する入力変数の集合を $\text{sup}(f_j)$ としたとき、(数 3) で表される T の値が最

小となるように前記多出力論理関数 $F(X)$ の要素の順序 π を決定する出力変数順序決定手段と、

前記各入力変数 x_i ($\in X$) 及び出力を表す変数 y_j ($\in Y$) の順序を、(数4)を満たす順序 P に決定する全変数順序決定手段と、

前記全変数順序決定手段で決定された順序 P に従って、特性関数二分決定グラフの節点データを生成し前記節点テーブル記憶手段に格納するBDD生成手段と、
を備えていることを特徴とする請求項1記載の論理回路合成装置。

【数3】

$$T = \sum_{k=0}^{m-1} \left| \bigcup_{l=0}^k \sup(f_{\pi[l]}) \right|$$

【数4】

$$P = \left(\sup(f_{\pi[0]}), y_{\pi[0]}, \sup(f_{\pi[1]}) - \sup(f_{\pi[0]}), y_{\pi[1]}, \sup(f_{\pi[2]}) - \left(\sum_{k=0}^1 \sup(f_{\pi[k]}) \right), y_{\pi[2]}, \right. \\ \left. \dots, \sup(f_{\pi[m-1]}) - \left(\sum_{k=0}^{m-2} \sup(f_{\pi[k]}) \right), y_{\pi[m-1]} \right)$$

【請求項3】

入力変数を $X=(x_1, \dots, x_n)$ ($n \in \text{自然数}$) とする多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して(数1)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2$, $m \in \text{自然数}$) は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフを格納する記憶手段であって、前記特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$) に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$) の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ の組からなる節点データが記憶されたものである節点テーブル記憶手段と、

前記多出力論理関数 $F(X)$ に対応する論理回路を構成するためのデータであるルックアップ・テーブルを格納するためのLUT記憶手段と、
を有する装置において、

前記特性関数二分決定グラフを分割する分割線の高さ lev を決定する第1ステップと、

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記高さ lev の分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものであって、出力を表す変数 y_r ($\in Y$) に関わる節点 v_j 及びその親節点 v_k の節点データについて、節点 v_j の枝 $e_0(v_j)$, $e_1(v_j)$ の一方 $e_a(v_j)$ が $\chi(X, Y)=0$ に関わる終端節点を特定する場合、当該節点 v_j の親節点 v_k の枝 $e_0(v_k)$, $e_1(v_k)$ のうち当該節点を特定する枝 $e_c(v_k)$ を、当該節点 v_j の枝 $e_a(v_j)$ 以外の枝 $e_b(v_j)$ に置き換える短絡除去処理を行う第2ステップと、

前記短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって高さ lev より高い高さにある非終端節点に属する枝のうち、高さ lev より低い高さの非終端節点の子節点を特定するものの個数を計数し(但し、同じ節点を特定するものは1つと数える。)
、その個数を高さ lev の分割線における幅 W として出力する第3ステップと、

前記BDD幅計測手段が出力する幅 W に基づき、(数2)の演算により中間変数の個数 u を算出する第4ステップと、

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記高さ lev の分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものについて、ルックアップ・テーブルを生成しそれをLUT記憶手段に格納する第5ステップと、

前記中間変数算出手段が算出する前記中間変数の個数 u と等しい制御入力数を有する二分木を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフ B_0 に属する非終端節点の節点データを、前記二分木を表す節点データ

で置き換えることにより、特性関数二分決定グラフを再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新する第6ステップと、
を複数回反復実行することにより、前記多出力論理関数 $F(X)$ に対応する論理回路を構成するためのデータであるルックアップ・テーブルを生成することを特徴とする論理回路合成方法。

【請求項4】

前記第1ステップから第6ステップを実行する前に、

前記多出力論理関数 $F(X)$ の要素をなす論理関数 $f_0(X), \dots, f_{m-1}(X)$ の順序を $\pi = (\pi[0], \dots, \pi[m-1])$ (但し、 $\pi[i]=j$ は、 f_j が i 番目であることを表す。)とし、論理関数 f_j ($\in F(X)$) が依存する入力変数の集合を $\text{sup}(f_j)$ としたとき、(数3)で表される T の値が最小となるように前記多出力論理関数 $F(X)$ の要素の順序 π を決定する出力変数順序決定ステップと、

前記各入力変数 x_i ($\in X$) 及び出力を表す変数 y_j ($\in Y$) の順序を、(数4)を満たす順序 P に決定する全変数順序決定ステップと、

前記順序 P に従って、特性関数二分決定グラフの節点データを生成し前記節点テーブル記憶手段に格納するBDD生成ステップと、
を実行することを特徴とする請求項3記載の論理回路合成方法。

【書類名】明細書

【発明の名称】論理回路合成装置及び論理回路合成方法

【技術分野】

【0001】

本発明は、ルックアップ・テーブル（以下、「LUT」という。）・カスケード論理回路やLUT型FPGA等において使用される、多出力論理関数 $f(X)$ に対応する論理回路を構成するためのデータであるLUTを生成するための論理回路合成技術に関するものである。

【背景技術】

【0002】

近年では、種々の電子回路の設計において、LUT型のフィールド・プログラマブル・ゲートアレイ（Field Programmable Gate Array：以下、「FPGA」という。）が広く使用されている（例えば、非特許文献1～3参照）。LUT型FPGAは、多数の再構成可能論理ブロック（Configurable Logic Block：以下、「CLB」という。）が格子状に配列され、各CLBの間に縦横に格子状に配線された複数の接続線（routing line：導線）を備えた構成からなる。各々の接続線の交点には、再構成可能なスイッチ・ブロック（Switch Block：以下「SB」という。）が配されており、各々の接続線の接続を自由に変更（再構成）することができる。また、各CLBと接続線とは、再構成可能な接続ブロック（Connection Block：以下、「CB」という。）により接続されている。そして、各接続線の終端には、FPGA外部との入出力を行う入出力部（I/O）が設けられている。各CLBは、内部に多入力1出力のLUTやマルチプレクサ（MUX）を1ないし複数個有し、このLUTやMUXにより論理演算が可能である。LUT型FPGAでは、SB、CB及びCLBの内部を再構成することにより、所望の多入力1出力論理関数が格納された複数のLUTの入出力を目的に応じてネットワーク状に順序接続し、様々な組み合わせ論理回路を実現することができる。

【0003】

一方、FPGAよりも高速な再構成可能論理回路を実現するものとして、LUTカスケード論理回路が提案されている（例えば、非特許文献4、5参照）。LUTカスケード論理回路においては、多入力多出力のLUTがカスケード状に接続された構成からなる。各LUTには、外部からの入力に加えて前段のLUTの出力が入力される。そして、最終段のLUTから1ないし複数の出力変数が出力される。演算を行おうとしている論理関数を複数の多出力論理関数に分解し、各多出力論理関数は各段のLUTに格納される。このようにして、LUTカスケード論理回路においては多出力論理関数の演算を行うことができる。

【0004】

上記のようなLUT型FPGAやLUTカスケード論理回路を、実際の論理回路設計に適用する場合には、まず、論理回路で実現しようとする論理関数（以下、「目的論理関数」という。）を複数の論理関数に関数分解して合成論理関数とする。ここで、合成論理関数とは、関数分解により得られる複数の論理関数（以下、「分解関数」という。）の結合論理からなる関数をいう。すなわち、目的論理関数を $f(X)$ （ $|X|$ は入力変数の集合）で表した場合に、この目的論理関数を、 $f(X_1, X_2) = g(h(X_1), X_2)$ （ $|X_1| \subset |X|$, $|X_2| \subset |X|$, $|X_1| \cap |X_2| = \phi$, $|X_1| \cup |X_2| = |X|$ ）の形に関数分解した場合、 $g(h(X_1), X_2)$ を g と h の合成論理関数という。

【0005】

尚、本明細書において、 X, Y と記す場合には、入力変数、出力変数のベクトル又は全順序集合（ordered set）を表し、 $|X|, |Y|$ と記すときは、入力変数、出力変数の非順序集合（unordered set）を表すものとする。

【0006】

上記関数分解により、2つの分解関数 $h(X_1)$, $g(h, X_2)$ が得られる。尚、かかる関数分解は常に可能であるとは限らないが、コンピュータ等で使用する制御回路や算術演算回路等

の多くの実用的関数は分解可能なものが多い(非特許文献6参照)。また、分解関数 $g(h, X_2)$ が更に分解可能であれば同様の関数分解を繰り返す。

【0007】

そして、各分解関数を別々のLUTとして実現し、各LUTを合成論理関数に従ってネットワーク状又はカスケード状に接続する。このようにして、目的論理関数をLUT型FPGAやLUTカスケード論理回路によって実現することができる。

【0008】

単一出力の目的論理関数を、上記関数分解の手法を用いて、LUTが結合した論理回路により実現することは比較的容易である(例えば、非特許文献4, 5参照)。

【0009】

一方、目的論理関数が多出力である場合において、目的論理関数をLUTが結合した論理回路により実現する論理合成技術としては、現在のところ以下の方法によるものが知られている。

- (1) 多端子二分決定グラフ(Multi-Terminal Binary Decision Diagram: MTBDD)を用いる方法(非特許文献7, 8参照)
- (2) 出力を幾つかのグループに分割して構成する方法(非特許文献7参照)
- (3) 分割理論を用いる方法(非特許文献9, 10参照)
- (4) 代入による方法(非特許文献11参照)
- (5) Hyper Functionを用いる方法(非特許文献12参照)
- (6) 非零出力符号化特性関数(Encoded Characteristic Function for Non-zero: ECFN)を用いた時分割実現による方法(非特許文献4, 5参照)
- (7) 以上のうちいくつかの方法の組み合わせによる方法(非特許文献11参照)

【非特許文献1】S.Brown, R.J.Francis, J.Rose, and Z.G.Vranesic, "Field-Programmable Gate Arrays", Kluwer Academic Publishers, 1992.

【非特許文献2】P.Chow, S.O.Seo, J.Rose, K.Chung, G.Paez-Monzon, and I.Rahardja, "The design of an SRAM-based field-programmable gate array ---Part I: Architecture," IEEE Transactions on VLSI Systems, vol.7, pp.191-197, June 1999.

【非特許文献3】Chow, P., S. Seo, J. Rose, K. Chung, G. Pez-Monzn and I. Rahardja. "The Design of an SRAM-Based Field-Programmable Gate Array, Part II: Circuit Design and Layout", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 7 No. 3, Sept. 1999, pp. 321-330.

【非特許文献4】笹尾勤, 松浦宗寛, 井口幸洋, "多出力関数のカスケード実現と再構成可能ハードウェアによる実現", 電子情報通信学会FTS研究会, FTS2001-8, pp. 57-64, 三重大学(2001-04).

【非特許文献5】T. Sasao, M. Matsuura, and Y. Iguchi, "A cascade realization of multiple-output function for reconfigurable hardware," International Workshop on Logic and Synthesis (IWLS01), Lake Tahoe, CA, June 12-15, 2001. pp. 225-230.

【非特許文献6】T.Sasao and M.Matsuura, "DECOMPOS: An integrated system for functional decomposition", 1998 International Workshop on Logic Synthesis, Lake Tahoe, June 1998.

【非特許文献7】Y-T.Lai, M.Pedram and S.B.K.Vrudhula, "BDD based decomposition of logic functions with application to FPGA synthesis", 30th ACM/IEEE Design Automation Conference, June 1993.

【非特許文献8】T.Sasao, "FPGA design by generalized functional decomposition", In Logic Synthesis and Optimization, Sasao ed., Kluwer Academic Publishers, pp.233-258, 1993.

【非特許文献9】C.Scholl and P.Molitor, "Communication based FPGA synthesis for multi-output Boolean functions", Asia and South Pacific Design Automation

n Conference, pp.279-287, Aug. 1995.

【非特許文献10】B.Wurth, K.Eckl, and K.Anterich, "Functional multiple-output decomposition: Theory and implicit algorithm", Design Automation Conf., p.54-59, June 1995.

【非特許文献11】H.Sawada, T.Suyama, and A.Nagoya, "Logic synthesis for look-up table based FPGAs using functional decomposition and support minimization", ICCAD, pp.353-359, Nov. 1995.

【非特許文献12】J.-H.R.Jian, J.-Y.Jou, and J.-D.Huang, "Compatible class encoding in hyper-function decomposition for FPGA synthesis", Design Automation Conference, pp.712-717, June 1998.

【非特許文献13】P.Ashar and S.Malik, "Fast functional simulation using branching programs", Proc. International Conference on Computer Aided Design, p.408-412, Nov. 1995.

【非特許文献14】C.Scholl, R.Drechsler, and B.Becker, "Functional simulation using binary decision diagram", ICCAD'97, pp.8-12, Nov. 1997.

【非特許文献15】A.Mishchenko and T.Sasao, "Logic Synthesis of LUT Cascades with Limited Rails", 電子情報通信学会VLSI設計技術研究会, VLD2002-9, 琵琶湖(2002-11)

【発明の開示】

【発明が解決しようとする課題】

【0010】

上記背景技術に挙げた(1)～(7)の方法による論理合成技術は、何れも多出力論理関数の論理合成の技術としては決定的な方法であるとはいえない。例えば、MTBDDを用いて多出力論理関数の関数分解を行うことは、理論上は可能である。しかしながら、 m 出力の論理関数の場合、最大で 2^m 個の終端節点が存在する。従って、MTBDDを用いて多出力論理関数の関数分解を行った場合には、多くの場合、列複雑度(後述の〔定義3〕参照)が増大し、実用的ではない。また、他の従来の方法に関しても同様なことがいえる。

【0011】

一方、比較器や加算器のように、すべての論理演算が終了しない段階においても、演算を進めるに従って、順次出力変数が決定していくような論理関数も多い。例えば、入力変数 $P=(p_1, p_2, p_3, p_4)$ ($P \in B^4$, $B=\{0, 1\}$) 及び $Q=(q_1, q_2, q_3, q_4)$ ($Q \in B^4$) を加算して出力変数 $R=(r_0, r_1, r_2, r_3, \text{cout})$ ($R \in B^5$) を出力するような加算器では、各桁の加算が終了した時点で、すべての桁の演算が終了する前にその桁に対応する出力変数が決定する。このように、すべての論理演算が終了しない段階で決定し出力される出力変数を「中間出力変数」という。

【0012】

目的論理関数が中間出力変数を有する場合、目的論理関数を実現するLUTネットワークやLUTカスケードにおいては、最終段以外の段のLUTにおいて中間出力変数を出力するように構成することにより、各LUTのサイズを小さくするとともに演算速度を向上させることが可能となる。

【0013】

しかしながら、上記背景技術に挙げた論理合成技術では、中間出力変数を有する目的論理関数に対して、中間出力を有するLUTネットワークやLUTカスケードを合成することは困難であった。

【0014】

そこで、本発明の目的は、一般的な多出力論理関数に対して汎用的に論理合成が可能であり、また、中間出力を有するLUTネットワークやLUTカスケードを合成することが可能な論理回路合成技術を提供することにある。

【課題を解決するための手段】

【0015】

以下、順を追って、まず本発明の基本的原理について説明し、その後、本発明の構成及び作用について説明する。

【0016】

〔1〕本発明の基本的原理

(1) 基本用語の定義

最初に、本明細書で用いる用語概念を明確化するために、以下の説明で用いられる基本用語について定義しておく。

【0017】

〔定義1〕

入力変数を $X=(x_1, x_2, \dots, x_n)$ とする。 X の変数の集合を $|X|$ で表す。 $|X_1| \cup |X_2| = |X|$ かつ $|X_1| \cap |X_2| = \phi$ のとき、 $X=(X_1, X_2)$ を X の分割 (partition) という。また、 X の変数の個数を $|X|$ で表す。

〔定義終わり〕

【0018】

〔定義2〕

論理関数 $f(X)$ に対して、 X の分割を $X=(X_1, X_2)$ とする。また、 $|X_1|=n_1$ 、 $|X_2|=n_2$ とする。このとき、 2^{n_1} 列 2^{n_2} 行の表で、各行、各列に 2 進符号のラベルを持ち、その要素が f の真値であるような表を、 f の分解表 (decomposition chart) という。このとき、 X_1 を束縛変数 (bound variable)、 X_2 を自由変数 (free variable) という。ここで、分解表の列、行は、それぞれ n_1 、 n_2 ビットのすべてのパターンを有する。

〔定義終わり〕

【0019】

〔定義3〕

分解表の異なる列パターンの個数を列複雑度 (column multiplicity) という。

〔定義終わり〕

【0020】

〔定義4〕

入力変数を $X=(x_1, x_2, \dots, x_n)$ とし、多出力論理関数を $f(X)=(f_0(X), f_1(X), \dots, f_{m-1}(X))$ とする。このとき、多出力論理関数 $f(X)$ の「特性関数」 $\chi(X, Y)$ を (数 1) により定義する (非特許文献 14 参照)。ここで、 $Y=(y_0, \dots, y_{m-1})$ は出力を表す変数である。

【0021】

【数 1】

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} (y_i \equiv f_i(X))$$

但し、 $Y=(y_0, \dots, y_{m-1})$ 、 $f(X)=(f_0(X), \dots, f_{m-1}(X))$ 、 $m \in \text{整数}$

〔定義終わり〕

【0022】

n 入力 m 出力関数の特性関数は $(n+m)$ 変数の二値論理関数であり、入力変数 x_i ($i=1, 2, \dots, n$) の他に、各出力 f_j ($j=0, 1, \dots, m-1$) に対して出力変数 y_j を用いる。 $B=\{0, 1\}$ とする。 $a \in B^n$ かつ $f(x)=(f_0(x), f_1(x), \dots, f_{m-1}(x)) \in B^m$ とする。いま、 $b \in B^m$ とすると、論理関数 $f(x)$ の特性関数 $\chi(a, b)$ の値は (数 2) のようになる。

【0023】

【数 2】

$$\chi(a, b) = \begin{cases} 1 & (b = f(a)) \\ 0 & (b \neq f(a)) \end{cases}$$

【0024】

〔定義 5〕

多出力論理関数 $f(X) = (f_0(X), f_1(X), \dots, f_{m-1}(X))$ の「特性関数二分決定グラフ」 (Binary Decision Diagram for Characteristic Function : BDD for CF) とは、多出力論理関数 $f(X)$ の特性関数 $\chi(X, Y)$ を表現する二分決定グラフ (Binary Decision Diagram : BDD) をいう。但し、二分決定グラフの変数は、根節点を最上位としたとき、変数 y_j は f_j に依存する入力変数 $x_i \in \text{sup}(f_j)$ よりも下位に置く。ここに、 $\text{sup}(f_j)$ は f_j の依存変数の集合を表す。(非特許文献 13, 14 参照)

〔定義終わり〕

【0025】

〔例 1〕

図 1 (a) の真理値表によって表現される多出力論理関数の特性関数二分決定グラフは、図 1 (b) により表される。ここで、○は変数節点を表し、□は特性関数値の節点を表す。

〔例終わり〕

【0026】

〔定義 6〕

特性関数二分決定グラフにおいて、出力を表す節点 y_j ($\in Y$) から出る枝 (edge) のうち、定数 0 に接続する枝を取り除き、節点 y_j の親節点 (parent node) から節点 y_j の定数 0 以外の子節点 (child node) へ直接枝を繋ぐ。この作業を、 y_j を表現する総ての節点に対して実行する操作を、出力変数 y_j の「短絡除去」 (shorten) という。

〔定義終わり〕

【0027】

具体的に図 2 を用いて短絡除去の説明をする。図 2 (a) に示したような出力を表す節点 y_j ($\in Y$) を考える。節点 y_j の親節点を z_p 、節点 y_j の子節点を z_q とする。まず、特性関数値 0 に接続する節点を除去すると、図 2 (b) のようになる。次に、節点 y_j の親節点 z_p から子節点 z_q へ直接枝を繋ぐことにより、図 2 (c) のようにグラフが変形される。節点 y_j に関する短絡除去では、以上のような操作を、特性関数二分決定グラフの中にある出力を表す節点 y_j のすべてについて行うものである。

【0028】

〔定義 7〕

二分決定グラフの高さ k での幅とは、変数 z_k と変数 z_{k+1} との間の枝の本数をいう。ここで、同じ節点に接続している枝は一つと数える。

〔定義終わり〕

【0029】

変数 z_i の節点の高さ (level) とは、二分決定グラフにおいて順序づけられた変数において、終端節点から数えた順番をいう。但し、終端節点の高さは 0 とする。すなわち、 $n+m$ 個の変数 $\{z_i ; i=1, \dots, n+m\}$ を有する二分決定グラフにおいて根節点から終端節点にかけて順序づけられた変数順序を $P = (z_{n+m}, z_{n+m-1}, \dots, z_1)$ とすると、変数 z_i の節点の高さは i となる。このとき、変数 z_i, z_j について $i > j$ のとき、変数 z_i は変数 z_j よりも高位 (high level) であるという。また、高さ i での幅は、変数 z_i と変数 z_{i+1} との間の枝の本数である。但し、同じ節点を特定するものは 1 つと数える。

【0030】

〔例 2〕

図 1 (b) の特性関数二分決定グラフにおいては、終端節点 (特性関数値がラベル付けられた節点) の高さが 0、変数 y_2 の高さが 1、変数 y_1 の高さが 2、変数 x_4 の高さが 3、…、変数 x_1 の高さが 7 となる。また、高さ 1 での幅は、変数 y_1 の節点から変数 y_2 の節点へ向かう枝の本数 (但し、終端節点 0 へ向かう枝は無視するものとする。) で 2 である。また、高さ 2 での幅は、変数 x_4 の節点から変数 y_1 の節点へ向かう枝の本数で 4 である。

〔例終わり〕

【0031】

(2) 目的論理関数の関数分解と LUT 回路の生成

本発明における LUT 論理回路の合成は、特性関数二分決定グラフの分割と短絡除去とを利用して行う。特性関数二分決定グラフを分割して 2 つの回路を構成した場合、両回路を接続する接続線数は、次の〔定理 1〕により算出することができる。

【0032】

〔定理 1〕

X_1, X_2 を入力変数の集合、 Y_1, Y_2 を出力変数の集合とする。特性関数二分決定グラフの変数順序を (X_1, Y_1, X_2, Y_2) とするとき、特性関数二分決定グラフの高さ $|X_2| + |Y_2|$ における幅を W とする。ここで、 W を数える際、出力を表現する変数から、定数 0 へ向かう枝は無視する。多出力論理関数を図 3 の回路で実現する場合、二つの回路 H と G の間の必要かつ十分な接続線数 u は、(数 3) により表される。

【0033】

〔数 3〕

$$u = \lceil \log_2 W \rceil$$

〔定理終わり〕

【0034】

(証明)

特性関数二分決定グラフの構成法から、出力関数 Y_1 と Y_2 が、図 3 の回路で表現できることは明らかである。特性関数二分決定グラフにおいて、 Y_1 の出力を表現する節点を短絡除去すると、 Y_1 以外の関数を表現する特性関数二分決定グラフが得られる。また、この操作によって、特性関数二分決定グラフの幅が増えることはない。高さ $|X_2| + |Y_2|$ における BDD の幅を W とする。いま、関数分解 $g(h(X_1), X_2)$ の分解表を考えると、その列複雑度は W に等しい。従って、回路 H と回路 G の間の配線数は、

【0035】

〔数 4〕

 $\lceil \log_2 W \rceil$ 〔本〕

だけあれば十分である。また、分解表の列複雑度は W なので、二つのブロック間の配線は少なくとも (数 4) の本数だけ必要である。

(証明終わり)

【0036】

特性関数二分決定グラフの分割及び短絡除去と上記〔定理 1〕とを利用して、次のように目的論理関数の関数分解が行われる。

【0037】

まず、目的論理関数 $f = (f_0, f_1, \dots, f_{m-1})$ の特性関数二分決定グラフの変数順序が (X_1, Y_1, X_2, Y_2) 、 $Y_1 = (y_0, y_1, \dots, y_{k-1})$ 、 $Y_2 = (y_k, y_{k+1}, \dots, y_{m-1})$ であるとする。この場合、 $y_j = f_j(X_1)$ ($j=0, 1, \dots, k-1$) は図 3 の回路 H で直接実現する。一方、高さ $|X_2| + |Y_2|$ における特性関数二分決定グラフの幅を W とする。 W 本の枝に u ビット (u は (数 3) により得られる数。) の異なる二進数を割り当てる。二つのブロック H, G 間を接続する枝が実現する関数を h_1, h_2, \dots, h_u とすると、ブロック G の出力関数 Y_2 は $(h_1, h_2, \dots, h_u, X_2)$ を入力変数とする論理関数として表現可能であり、その特性関数二分決定グラフは図 4 のようになる。

【0038】

〔例 3〕

本例では、2 つの 2 ビットの 2 進数を加算する回路 (ADR 2) を、中間出力を有する関数分解を用いて実現する場合について説明する。ADR 2 の入出力の関係は (数 5) のように定義できる。これより、 s_0, s_1, s_2 は、それぞれ (数 6) により表される。また、真理値表は図 5 (a) により表される。

【0039】

【数5】

$$\begin{array}{r}
 \\
 \\
 \hline
 s_2 s_0
 \end{array}$$

【0040】

【数6】

$$s_0 = a_0 \oplus b_0$$

$$s_1 = a_0 b_0 \oplus (a_1 \oplus b_1)$$

$$s_2 = a_0 b_0 (a_1 \vee b_1) \vee a_1 b_1$$

【0041】

変数の分割として、 $X_1=(a_0, b_0)$, $Y_1=(s_0)$, $X_2=(a_1, b_1)$, $Y_2=(s_1, s_2)$ とおく。このとき、変数順序は、 $(X_1, Y_1, X_2, Y_2)=(a_0, b_0, s_0, a_1, b_1, s_1, s_2)$ となる。従って、ADR2の特性関数二分決定グラフは、図5(b)により表される。 $Z=(Z_A, Z_B)$, $Z_A=(X_1, Y_1)$, $Z_B=(X_2, Y_2)$ のように分割し、 Z_A を図3の回路H、 Z_B を図3の回路Gにより構成する。このとき、高さ $|Z_B|$ での特性関数二分決定グラフの幅Wは2となる。従って、回路Hと回路Gとを連結する線は、(数3)より、1本あればよい。出力 s_0 は、 X_1 のみの関数として表現することが可能である。

【0042】

また、変数組 Z_A により構成される特性関数二分決定グラフは、図6(a)のようになる。これは、図5の特性関数二分決定グラフの分割線よりも根節点側(上側)の部分を取り切ったものである。尚、図6(a)では特性関数値0につながる枝は省略してある。回路Hと回路Gとを連結する線は1本なので、図6(a)のグラフの終端節点には、1ビットの中間変数 h_1 を導入して、各終端節点に割り当てる。なお、変数 h_1 への符号の割り当ては任意である。この特性関数二分決定グラフは、容易に多端子二分決定グラフ(MTBDD)に変換することができる。図6(a)のグラフを変換することにより得られるMTBDDは、図6(b)のようになる。図6(b)のMTBDDから回路Hに相当するLUTを図6(c)のように生成することができる。

【0043】

次に、図5の特性関数二分決定グラフについて、先に導入した新しい中間変数 h_1 を用いて、図5(b)に示した特性関数二分決定グラフの分割線よりも上部を、 h_1 を入力変数とする決定木(decision tree)で置き換える。これにより、特性関数二分決定グラフは図7(a)のように変形される。尚、図7(a)でも特性関数値0につながる枝は必要ないので省略してある。この特性関数二分決定グラフは、容易にMTBDDに変換することができる。図7(a)のグラフを変換することにより得られるMTBDDは、図7(b)のようになる。更に、図7(b)のMTBDDから回路Gに相当するLUTを図8(a)のように生成することができる。従って、結局、図8(b)に示したようなLUTカスケードを構成することができる。

【例終わり】

【0044】

〔2〕本発明の構成及び作用

本発明に係る論理回路合成装置の第1の構成は、入力変数を $X=(x_1, \dots, x_n)$ ($n \in \text{自然数}$)とする多出力論理関数 $f(X)=(f_0(X), \dots, f_{n-1}(X))$ に対して(数1)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{n-1})$ ($m \geq 2$, $m \in \text{自然数}$)は $f(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフ(Binary Decision Diagram for Characteristic Function: BDD for CF)を格納する記憶手段であって、前記特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先

の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ の組からなる節点データが記憶されたものである節点テーブル記憶手段と、

前記多出力論理関数 $f(X)$ に対応する論理回路を構成するためのデータであるルックアップ・テーブル (Look-Up Table: LUT) を格納するための LUT 記憶手段と、

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを所定の高さ lev の分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものであって、出力を表す変数 y_r ($\in Y$) に関わる節点 v_j 及びその親節点 v_k の節点データについて、節点 v_j の枝 $e_0(v_j)$, $e_1(v_j)$ の一方 $e_a(v_j)$ が $\chi(X, Y)=0$ に関わる終端節点を特定する場合、当該節点 v_j の親節点 v_k の枝 $e_0(v_k)$, $e_1(v_k)$ のうち当該節点を特定する枝 $e_c(v_k)$ を、当該節点 v_j の枝 $e_a(v_j)$ 以外の枝 $e_b(v_j)$ に置き換える短絡除去処理を行う短絡除去手段と、

前記短絡除去手段により短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって高さ lev より高い高さにある非終端節点に属する枝のうち、高さ lev より低い高さの非終端節点の子節点を特定するものの個数を計数し (但し、同じ節点を特定するものは1つと数える。)、その個数を高さ lev の分割線における幅 W として出力する BDD 幅計測手段と、

前記 BDD 幅計測手段が出力する幅 W に基づき、(数3) の演算により中間変数の個数 u を算出する中間変数算出手段と、

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを所定の高さ lev の分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものについて、ルックアップ・テーブルを生成しそれを LUT 記憶手段に格納する LUT 生成手段と、

前記中間変数算出手段が算出する前記中間変数の個数 u と等しい制御入力数を有する二分木 (binary tree) を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフ B_0 に属する非終端節点の節点データを、前記二分木を表す節点データで置き換えることにより、特性関数二分決定グラフを再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新する BDD 再構成手段と、を備えていることを特徴とする。

【0045】

この構成によれば、

(a) まず、論理回路合成を行う目的論理関数の特性関数二分決定グラフの節点データを節点テーブル記憶手段に格納しておく。短絡除去手段は、節点テーブル記憶手段に格納された節点データで構成される特性関数二分決定グラフについて、所定の高さ lev の分割線に対して根節点を含む側の部分グラフ B_0 について短絡除去処理を行う。短絡除去処理については、上記「[1] 本発明の基本的原理」の欄において説明した通りである。BDD 幅計測手段は、上記短絡除去処理が行われた結果得られる特性関数二分決定グラフについて、上記分割線における幅 W を計測する。そして、中間変数算出手段は、(数3) によって中間変数の個数 u を算出する。

【0046】

(b) 次に、LUT 生成手段は、節点テーブル記憶手段に格納された節点データで構成される特性関数二分決定グラフについて、所定の高さ lev の分割線に対して根節点を含む側の部分グラフ B_0 について、ルックアップ・テーブルを生成する。この生成方法の原理については、上記「[1] 本発明の基本的原理」の欄において説明した通りである。この LUT の出力は、部分グラフ B_0 に属する出力変数 $Y_0=(y_0, \dots, y_{k-1})$ 及び u 個の中間変数 $H=(h_1, \dots, h_u)$ となる。生成されたルックアップ・テーブルは、LUT 記憶手段に格納される。中間変数 H の節点に対しては、適当な符号を割り当てることができる。

【0047】

(c) 最後に、BDD 再構成手段は、 u 個の制御入力数を有する二分木を生成し、各制御入力数に対して上記中間変数 $H=(h_1, \dots, h_u)$ を対応させる。そして、BDD 再構成手段は、

この処理により再構成された特性関数二分決定グラフの各節点の節点データを節点テーブル記憶手段に格納し、節点テーブル記憶手段が保持する特性関数二分決定グラフを更新する。

【0048】

以上のような(a)～(c)の処理を繰り返せば、目的論理関数は複数の部分関数に関数分解され、LUTカスケード論理回路が構成される。また、LUT生成手段により生成されるルックアップ・テーブルを特性関数二分決定グラフにして、更に同様の処理によって関数分解を行うことによって、LUTネットワーク論理回路が構成される。

【0049】

本構成による論理回路合成装置では、分割線の位置を、部分グラフ B_0 の大きさが過度に大きくならないように適度な高さに決めることで、部分グラフ B_0 に関する変数を束縛変数とした分解表の列複雑度が極度に増大することを防止することが可能である。従って、比較的少容量のメモリを用いて論理回路合成装置を実現することが可能である。また、部分グラフ B_0 に関する変数を束縛変数とした分解表の列複雑度を抑えることができるため、計算処理量やメモリ・アクセス回数を少なくすることが可能であり、論理関数分解処理の演算処理速度を高速化することができる。

【0050】

本発明に係る論理回路合成装置の第2の構成は、前記第1の構成において、前記多出力論理関数 $f(X)$ の要素をなす論理関数 $f_0(X), \dots, f_{m-1}(X)$ の順序を $\pi = (\pi[0], \dots, \pi[m-1])$ (但し、 $\pi[i]=j$ は、 f_j が i 番目であることを表す。)とし、論理関数 $f_j (\in f(X))$ が依存する入力変数の集合を $\text{sup}(f_j)$ としたとき、(数7)で表される T の値が最小となるように前記多出力論理関数 $f(X)$ の要素の順序 π を決定する出力変数順序決定手段と、

前記各入力変数 $x_i (\in X)$ 及び出力を表す変数 $y_j (\in Y)$ の順序を、(数8)を満たす順序 P に決定する全変数順序決定手段と、

前記全変数順序決定手段で決定された順序 P に従って、特性関数二分決定グラフの節点データを生成し前記節点テーブル記憶手段に格納するBDD生成手段と、を備えていることを特徴とする。

【0051】

【数7】

$$T = \sum_{k=0}^{m-1} \left| \bigcup_{l=0}^k \text{sup}(f_{\pi[l]}) \right|$$

【0052】

【数8】

$$P = \left(\text{sup}(f_{\pi[0]}), y_{\pi[0]}, \text{sup}(f_{\pi[1]}) - \text{sup}(f_{\pi[0]}), y_{\pi[1]}, \text{sup}(f_{\pi[2]}) - \left(\sum_{k=0}^1 \text{sup}(f_{\pi[k]}) \right), y_{\pi[2]}, \right. \\ \left. \dots, \text{sup}(f_{\pi[m-1]}) - \left(\sum_{k=0}^{m-2} \text{sup}(f_{\pi[k]}) \right), y_{\pi[m-1]} \right)$$

【0053】

実用的な多出力論理関数の特性関数二分決定グラフは、通常はかなりの大きさとなる。そのため、出力を分割する方法が必要である。そこで、上記構成によれば、出力変数順序決定手段が、論理関数 $f_0(X), \dots, f_{m-1}(X)$ の順序を(数7)で表される T の値が最小となるように決定することにより、BDD生成手段が生成する特性関数二分決定グラフの節点データの数を削減できる。特性関数二分決定グラフにおいて、各出力変数 $y_j = f_j(X)$ に対して、その出力変数が依存する入力変数が、その出力変数 y_j よりも上位に置かれるため、依存変数になるべく増えないような順序に並べられ、一つずつ出力を増やしながら特性関数二分決定グラフが作られる。このようにして、出力を最適に分割する。このとき、特性関数二分決定グラフの節点データの数が削減できるので、その後の論理回路合成に要する処理

時間が短縮され、高速な論理回路合成が可能となる。

【0054】

本発明に係る論理回路合成方法の第1の構成は、入力変数を $X=(x_1, \dots, x_n)$ ($n \in \text{自然数}$) とする多出力論理関数 $f(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して (数1) により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2, m \in \text{自然数}$) は $f(X)$ の出力を表す変数。) を表現する特性関数二分決定グラフを格納する記憶手段であって、前記特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$) に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$) の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ の組からなる節点データが記憶されたものである節点テーブル記憶手段と、

前記多出力論理関数 $f(X)$ に対応する論理回路を構成するためのデータであるルックアップ・テーブルを格納するためのLUT記憶手段と、

を有する装置において、

(1) 前記特性関数二分決定グラフを分割する分割線の高さ lev を決定する第1ステップと、

(2) 前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記高さ lev の分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものであって、出力を表す変数 y_r ($\in Y$) に関わる節点 v_j 及びその親節点 v_k の節点データについて、節点 v_j の枝 $e_0(v_j)$, $e_1(v_j)$ の一方 $e_a(v_j)$ が $\chi(X, Y)=0$ に関わる終端節点を特定する場合、当該節点 v_j の親節点 v_k の枝 $e_0(v_k)$, $e_1(v_k)$ のうち当該節点を特定する枝 $e_c(v_k)$ を、当該節点 v_j の枝 $e_a(v_j)$ 以外の枝 $e_b(v_j)$ に置き換える短絡除去処理を行う第2ステップと、

(3) 前記短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって高さ lev より高い高さにある非終端節点に属する枝のうち、高さ lev より低い高さの非終端節点の子節点を特定するものの個数を計数し (但し、同じ節点を特定するものは1つと数える。) 、その個数を高さ lev の分割線における幅 W として出力する第3ステップと、

(4) 前記BDD幅計測手段が出力する幅 W に基づき、(数3) の演算により中間変数の個数 u を算出する第4ステップと、

(5) 前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記高さ lev の分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものについて、ルックアップ・テーブルを生成しそれをLUT記憶手段に格納する第5ステップと、

(6) 前記中間変数算出手段が算出する前記中間変数の個数 u と等しい制御入力数を有する二分木を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフ B_0 に属する非終端節点の節点データを、前記二分木を表す節点データで置き換えることにより、特性関数二分決定グラフを再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新する第6ステップと、

を複数回反復実行することにより、前記多出力論理関数 $f(X)$ に対応する論理回路を構成するためのデータであるルックアップ・テーブルを生成することを特徴とする。

【0055】

本発明に係る論理回路合成方法の第2の構成は、前記第1の構成において、前記第1ステップから第6ステップを実行する前に、

前記多出力論理関数 $f(X)$ の要素をなす論理関数 $f_0(X), \dots, f_{n-1}(X)$ の順序を $\pi = (\pi[0], \dots, \pi[m-1])$ (但し、 $\pi[i]=j$ は、 f_j が i 番目であることを表す。) とし、論理関数 f_j ($\in f(X)$) が依存する入力変数の集合を $\text{sup}(f_j)$ としたとき、(数7) で表される T の値が最小となるように前記多出力論理関数 $f(X)$ の要素の順序 π を決定する出力変数順序決定ステップと、

前記各入力変数 x_i ($\in X$) 及び出力を表す変数 y_j ($\in Y$) の順序を、(数8) を満たす順序 P に決定する全変数順序決定ステップと、

前記順序Pに従って、特性関数二分決定グラフの節点データを生成し前記節点テーブル記憶手段に格納するBDD生成ステップと、
を実行することを特徴とする。

【発明の効果】

【0056】

以上のように、本発明によれば、特性関数二分決定グラフを分割する分割線の位置を、部分グラフ B_0 の大きさが過度に大きくならないように適度な高さに決めることで、部分グラフ B_0 に関する変数を束縛変数とする分解表の列複雑度が極度に増大するのを防止することが可能となる。その結果、比較的小容量のメモリを用いて論理回路合成を実現でき、また、論理関数分解処理の演算処理速度を高速化することができる。従って、現在まで有効な論理回路合成手段が知られていなかった、多出力論理関数の分解によるLUT論理回路の生成を、現実的なハードウェアを使用して短時間で実行することが可能となる。また、本発明は、任意の多出力論理関数のLUT論理回路生成に対して汎用的に適用可能である。

【発明を実施するための最良の形態】

【0057】

以下、本発明を実施するための最良の形態について、図面を参照しながら説明する。

【実施例1】

【0058】

図9は本発明の実施例1に係る論理回路合成装置及びその周辺装置の構成を表す図である。実施例1に係る論理回路合成装置1は、論理仕様記憶手段2に格納された論理回路の論理仕様から、LUT論理回路を合成し、出力装置4に出力する。論理仕様記憶手段2には、入力装置3により、目的論理関数がネットリストなど論理仕様データとして格納される。

【0059】

論理回路合成装置1は、出力変数順序決定手段5、全変数順序決定手段6、BDD生成手段7、節点テーブル記憶手段8、変数順序最適化手段9、分割線決定手段10、短絡除去手段11、BDD幅計測手段12、中間変数算出手段13、LUT生成手段14、BDD再構成手段15、及びLUT記憶手段16を備えた構成からなる。

【0060】

出力変数順序決定手段5は、多出力論理関数 $f(X)$ の要素の順序 $\pi = (\pi[0], \dots, \pi[m-1])$ を決定する。全変数順序決定手段6は、各入力変数 x_i ($\in X$) 及び出力を表す変数 y_j ($\in Y$) の順序を、所定の順序Pに決定する。BDD生成手段7は、全変数順序決定手段6で決定された順序Pに従って、特性関数二分決定グラフの節点テーブルを生成する。生成された節点テーブルは、節点テーブル記憶手段8に格納される。変数順序最適化手段9は、節点テーブル記憶手段8に格納された節点テーブルの変数順序を更に最適化する。

【0061】

分割線決定手段10は、節点テーブル記憶手段8に格納された特性関数二分決定グラフの節点データに対して、当該特性関数二分決定グラフを分割する分割線の高さlevの決定を行う。短絡除去手段11は、節点テーブル記憶手段8に格納された特性関数二分決定グラフに対し、高さlevの分割線より上の部分グラフの短絡除去処理を行う。BDD幅計測手段12は、短絡除去処理がされた特性関数二分決定グラフの分割線における幅Wとして出力する。中間変数算出手段13は、BDD幅計測手段12が出力する幅Wに基づき中間変数の個数uを算出する。LUT生成手段14は、分割線において2つの部分グラフ B_0, B_1 変数の個数uを算出する。LUT生成手段14は、分割線において2つの部分グラフ B_0, B_1 に分割したときの根節点を含む側の部分グラフ B_0 に属するものについて、LUTを生成しそれをLUT記憶手段16に格納する。BDD再構成手段15は、部分グラフ B_0 を中間変数の個数uと等しい制御入力数を有する二分木に置き換えて、特性関数二分決定グラフを再構成し、節点テーブル記憶手段8を更新する。

【0062】

以上のように構成された本実施例に係る論理回路合成装置1について、以下その動作を

説明する。

【0063】

図10は実施例1における論理回路合成方法の全体の流れを示すフローチャートである。全体的な処理の流れでは、最初に、出力変数順序決定手段5と全変数順序決定手段6が、論理仕様記憶手段2に格納された論理仕様を読み出す。そして、読み出された論理仕様から、入力変数Xと出力変数Yとを抽出し、これら各変数の初期順序の決定を行う(S1)。次に、BDD生成手段7は、論理仕様記憶手段2から論理仕様データを読み出し、上記決定された変数の初期順序に従って特性関数二分決定グラフの節点テーブルを生成する。生成された節点テーブルは、節点テーブル記憶手段8に格納される(S2)。次に、変数順序最適化手段9は、節点テーブル記憶手段8に格納された節点テーブルにおいて、特性関数二分決定グラフの幅の総和がより小さくなるように、更に変数順序の入れ替えを行い、変数順序を最適化する(S3)。このとき、変数順序の入れ替えは、出力変数の順序集合 $Y=(y_0, y_1, \dots, y_{m-1})$ における出力変数 y_j ($j=0, 1, \dots, m-1$) に対して、その出力変数 y_j が依存する入力変数 x_i ($i \in \text{sup}(f_j)$) は、出力変数 y_j よりも常に高位であるという条件のもとで、変数の入れ替えが行われる。最後に、節点テーブル記憶手段8に格納された特性関数二分決定グラフの節点テーブルに基づいて、LUT論理回路の合成が行われる(S4)。以下、上記各段階での処理について詳述する。

【0064】

(1) 変数X, Yの初期順序の決定

図11は変数順序の決定処理の流れを表すフローチャートである。まず、出力変数順序決定手段5は、内部変数 i, j, renew を0に初期化する(S10)。 i, j は置換する変数のインデックスを表す内部変数であり、 renew は順序の置換が行われたことを検出するための更新フラグである。また、出力変数順序決定手段5は、出力変数Yの変数順序を表す m 個の要素を持つ順序 $\pi_0=(\pi_0[0], \pi_0[1], \dots, \pi_0[m-1])$ を $(0, 1, \dots, m-1)$ に初期化する(S2)。順序 π_0 は、出力変数Yの変数順序を表す配列である。出力変数の順序は、順序 π_0 により、(数9)のように順序づけされる。

【0065】

【数9】

$$Y = (y_{\pi_0[0]}, y_{\pi_0[1]}, \dots, y_{\pi_0[m-1]})$$

【0066】

次に、出力変数順序決定手段5は、(数10)の演算を行うことにより、順序 π_0 に対する出力変数順序評価関数を算出し、これを変数 T_{\min} に格納する(S12)。ここで、出力変数順序評価関数とは、(数10)の右辺の関数である。この出力変数順序評価関数は、各出力変数Yの順序 π_0 が、依存変数の数の少ない順に出力変数Yを順序づけするものであるときに最小となる。従って、出力変数順序評価関数を最小化することにより、出力変数Yの順序は最適化されることとなる。また、 $\text{sup}(f_j)$ は、論理関数 f_j ($f_j \in f(X)$) が依存する入力変数の集合を表す。例えば、 $f_j = f_j(x_1, x_2, x_5, x_{10})$ であれば、 $|\text{sup}(f_j)|=4$ である。また、 $|\text{sup}(f_j)|$ は $\text{sup}(f_j)$ の変数の個数である([定義1]参照)。

【0067】

【数10】

$$T_{\min} = \sum_{k=0}^{m-1} \left| \bigcup_{l=0}^k \text{sup}(f_{\pi_0[l]}) \right|$$

【0068】

次に、出力変数順序決定手段5は、 $i \neq j$ であるかどうかを判定する(S13)。 $i=j$ の場合には、ステップS19に行く。これは、 $i=j$ の場合には置換が行われないので、 $i=j$ の場合を除外する必要があるからである。

【0069】

ステップ S 1 3 において、 $i \neq j$ であれば、まず、変数置換後の順序 π_1 を初期順序 π_0 に初期化した後 (S 1 4)、順序 π_1 の要素 $\pi_1[i]$ と要素 $\pi_1[j]$ との置換を行う (S 1 5)。そして、出力変数順序決定手段 5 は、(数 1 1) の演算を行うことにより、順序 π_1 に対する出力変数順序評価関数 T を算出する (S 1 6)。

【0070】

【数 1 1】

$$T = \sum_{k=0}^{m-1} \left| \bigcup_{l=0}^k \sup(f_{\pi_1[l]}) \right|$$

【0071】

ここで、置換後の出力変数順序評価関数 T が置換前の出力変数順序評価関数 T_{\min} よりも小さい ($T < T_{\min}$) 場合には (S 1 7)、順序 π_1 のほうが出力変数の順序としてはより適当であると判断されるため、出力変数順序決定手段 5 は、順序 π_0 を π_1 に更新し、 T_{\min} を T とする。そして、順序 π_0 の更新がされたことを表す更新フラグ renew を 1 とする (S 1 8)。

【0072】

次に、 $j < m-1$ であれば (S 1 9)、 j を 1 だけインクリメントして (S 2 0)、ステップ S 1 3 に戻り、上記出力変数の順序の最適化処理を繰り返す。

【0073】

ステップ S 1 8 で $j = m-1$ ならば、 i 番目の出力変数の順序を固定したときの順序の入れ替えに対する評価がすべて終了したことになるので、 j を 0 に初期化する (S 2 1)。

【0074】

次に、 $i < m-1$ であれば (S 2 2)、次の出力変数の順序を固定したときの順序の入れ替えに対する評価を行うべく、 i を 1 だけインクリメントして (S 2 3)、ステップ S 1 3 に戻る。

【0075】

ステップ S 2 2 において、 $i = m-1$ の場合、 i 番目の出力変数の順序を固定した場合についての評価は一通り終了したことになる。但し、順序入れ替えの総数は、 $m!$ 通りであるが、この段階では、順序入れ替えの評価は $m \times (m-1)$ 通りしか行われていない。そこで、残りの場合についても評価するかどうかを判断するため、上記 $m \times (m-1)$ 通りの評価において順序 π_0 の更新が行われたかどうかを判定する。すなわち、変数 renew が 1 であるかどうかを参照する (S 2 4)。ここで、 $\text{renew} = 1$ であれば、 i を 0、 renew を 0 に初期化して (S 2 5)、再びステップ S 1 3 の処理に戻る。尚、ステップ S 2 4 で $\text{renew} = 0$ であれば、終了する。

【0076】

以上のようにして、出力変数 Y の順序 π_0 は、出力変数順序評価関数 T_{\min} の値が極小化する順序に最適化される。出力変数順序決定手段 5 は、出力変数 Y の順序を順序 π_0 に並べ替えて出力する。

【0077】

尚、上記処理では、順序入れ替えの評価は必ずしも $m!$ 回行われないので、必ずしも出力変数順序評価関数 T_{\min} の値が最小となるように最適化されるとは限らないことに注意しておく。 $m!$ 回のすべてについての評価を行わないこととしたのは、処理時間を短縮するためである。しかしながら、上記変数並べ替えのアルゴリズムは一例であり、計算処理速度が十分速い場合や、出力変数及び入力変数の数が比較的少ない場合には、 $m!$ 回のすべてについて総当たりで評価を行うことにより、出力変数順序評価関数 T_{\min} の値が最小となるように最適化してもよい。

【0078】

出力変数の順序が決定された後、全変数順序決定手段 6 は、入力変数 X 及び出力変数 Y の順序を (数 1 2) の順序 P に従って決定する。

【0079】

【数12】

$$P = \left(\sup(f_{\pi[0]}), y_{\pi[0]}, \sup(f_{\pi[1]} - \sup(f_{\pi[0]}), y_{\pi[1]}, \sup(f_{\pi[2]} - \left(\sum_{k=0}^1 \sup(f_{\pi[k]}) \right), y_{\pi[2]}, \right. \\ \left. \dots, \sup(f_{\pi[m-1]} - \left(\sum_{k=0}^{m-2} \sup(f_{\pi[k]}) \right), y_{\pi[m-1]} \right)$$

ここで、(数12)の条件を満たす限りにおいては、入力変数Xの順序の決め方は任意である。したがって、ここでは、出力変数

【0080】

【数13】

$$y_{\pi[j]} \quad (j = 0, 1, \dots, m-1)$$

の前にある入力変数の集合

【0081】

【数14】

$$(x_{\theta[p]}, x_{\theta[p+1]}, \dots, x_{\theta[p+q-1]}) = \sup(f_{\pi[j]}) - \left(\sum_{k=0}^{j-1} \sup(f_{\pi[k]}) \right)$$

の要素間の順序については、 x_i のインデックス i が小さい順に順序づけることとする。尚、(数14)において $(\theta[1], \theta[2], \dots, \theta[p], \theta[p+1], \dots, \theta[p+q-1], \dots, \theta[n])$ は、入力変数Xの順序を表し、 $\theta[k]=i$ の場合、 x_i は入力変数の k 番目に位置することを表す。

【0082】

(2) 特性関数二分決定グラフの節点テーブルの作成

変数X, Yの初期順序Pの決定がされた後、BDD生成手段7は、決定された初期順序Pに従って、論理仕様記憶手段2に記憶された論理仕様に従って、特性関数二分決定グラフの節点テーブルを生成する。この節点テーブルは、節点テーブル記憶手段8に格納される。尚、論理仕様から特性関数二分決定グラフを生成するアルゴリズムに関しては、すでに種々のアルゴリズムが公知であるため、ここでは説明を省略する。

【0083】

図12は節点テーブルのデータ構造を表す図である。1つの節点に対応する節点データは、(変数ラベル, 0枝の子節点のアドレス, 1枝の子節点のアドレス)の3つのデータ組からなる。変数ラベルには、各節点に対応する入力変数又は出力変数を特定する符号が格納される。通常は、計算処理の便宜のため、 $m+n$ 個の各変数 $x_1, \dots, x_n, y_0, \dots, y_{m-1}$ には2進数の符号が割り当てられるが、ここでは説明の都合上、変数 $x_1, \dots, x_n, y_0, \dots, y_{m-1}$ の記号をそのまま使用する。「0枝」(0-edge)とは、その節点に対応する変数の値(リテラル)が0であるときの節点間の遷移を表す枝(edge)をいい、「1枝」(1-edge)とは、その節点に対応する変数の値(リテラル)が1であるときの節点間の遷移を表す枝をいう。「0枝の子節点」(「1枝の子節点」)とは、その節点に対して0枝(1枝)に沿って遷移した先の節点をいう。「節点のアドレス」とは、節点テーブル記憶手段8において、その節点データが格納されている記憶装置の論理アドレスをいう。

【0084】

従って、ある節点に対応する変数の値が与えられた場合に、その変数値に対する遷移先の節点を参照する場合には、変数値0, 1に応じて0枝の子節点のアドレス, 1枝の子節点のアドレスを参照すれば、特性関数二分決定グラフの径路(path)を辿ることができる。

【0085】

【例4】

入力変数 $X=(x_1, x_2, x_3)$ に対して (数 15) により表される多出力論理関数 $f(X)$ について、変数順序 P は、上記アルゴリズムによって $P=(x_1, x_2, y_0, x_3, y_1)$ となる。

【0086】

【数 15】

$$\begin{aligned} F(X) &= (f_0(X), f_1(X)) \\ &= (x_1x_2, x_1 \vee x_3) \end{aligned}$$

【0087】

ここで、(数 15) の多出力論理関数 $f(X)$ の特性関数二分決定グラフは、図 13 (b) により表される。従って、図 13 (b) の特性関数二分決定グラフに対応する節点テーブルは、図 13 (a) のようになる。ここで、終端節点のうち、特性関数値 0 に対応する終端節点にはアドレス 0、特性関数値 1 に対応する終端節点にはアドレス 1 が割り当てられる。

〔例終わり〕

【0088】

(3) 変数 X, Y の順序の最適化

次に、変数 X, Y が初期順序により順序づけられた特性関数二分決定グラフの節点テーブルが節点テーブル記憶手段 8 に格納されると、変数順序最適化手段 9 は、変数 X, Y の間の順序の入れ替えを実行し、変数 X, Y 間の順序の最適化を行う。このとき、変数 X の順序の入れ替えには (数 16) の条件 (すなわち、「集合 $\sup(f_j)$ 」に属するすべての入力変数 x_i は y_j よりも高位である。」という条件) が課される。

【0089】

【数 16】

$$x_i \succ y_j \quad (\forall x_i \in \sup(f_j))$$

(但し、 $z_p \succ z_q$ は、変数 z_p の高さが変数 z_q の高さよりも高い (z_p は z_q より高位である) ことを表す。)

【0090】

〔例 5〕

(数 15) の多出力論理関数 $f(x)$ で表現される論理仕様に対して、上記の変数順序 $P=(x_1, x_2, y_0, x_3, y_1)$ による特性関数二分決定グラフの節点テーブルが、図 13 のように生成され、節点テーブル記憶手段 8 に格納されているとする。変数順序最適化手段 9 は、この節点テーブルに対して、上記条件のもとでの変数順序 P の入れ替えを行う。そして、入れ替え後の変数順序 P' により、特性関数二分決定グラフの再構築を行い、その結果、特性関数二分決定グラフの幅の総和が小さくなった場合にのみ、変数順序 P' の節点テーブルで節点テーブル記憶手段 8 の内容を更新する。

【0091】

例えば、変数順序最適化手段 9 が上記の変数順序 P を順序 $P'=(x_1, x_2, x_3, y_0, y_1)$ のように入れ替えを行った場合、特性関数二分決定グラフは図 14 (b) のようになる。また、この特性関数二分決定グラフに対応する節点テーブルは図 14 (b) のようになる。図 14 (a) の節点テーブルの大きさと図 13 (a) の節点テーブルの大きさは同じである。従って、この場合、節点テーブル記憶手段 8 の内容の更新は行われない。

【0092】

このような変数順序の入れ替えによる節点テーブルの再構築を総当たり (又は、適当な条件を課した試行) で試みることにより、変数順序の最適化が行われる。尚、(数 15) の多出力論理関数の場合、図 13 の特性関数二分決定グラフが最小であるため、節点テーブル記憶手段 8 内の節点テーブルの更新は行われない。

〔例終わり〕

【0093】

(4) LUTカスケード論理回路の合成

以上のように、最適化された特性関数二分決定グラフの節点テーブルが節点テーブル記憶手段 8 に格納されると、続いて LUT カスケード論理回路の合成処理が行われる。そこで、次に、LUT カスケード論理回路の合成処理について説明する。

【0094】

図 15 は LUT カスケード論理回路の合成処理の流れを表すフローチャートである。論理回路合成装置 1 は、分割線の高さを表す内部変数 i ($i-1$ が分割線の高さを表す。) を $n+m+1$ (n は入力変数 X の大きさ ($n=|X|$)、 m は出力変数 Y の大きさ ($m=|Y|$)) とする。また、中間変数の集合 H を空集合 ϕ とする。また、分割線より高位の変数の集合 Z_a を空集合 ϕ とする。また、関数分解を行う特性関数二分決定グラフ $B_{CF}^{current}$ を関数 f の特性関数二分決定グラフ $B_{CF}(f)$ とし、 $B_{CF}^{current}$ に含まれる変数の集合 Z_t を全変数の集合 $Z (=X \cup Y)$ に初期化する (S30)。

【0095】

次に、分割線決定手段 10 は、変数 i を 1 だけ減少する (S31)。すなわち、分割線の高さを 1 だけ下げる。そして、集合 Z_{temp} を $Z_a \cup \{z_i\}$ とする (S32)。ここで、 $\{z_i\}$ は高さ i の変数 z_i の集合を表す。すなわち、分割線を下げたので、集合 $\{z_i\}$ を分割線よりも高位の変数の集合 Z_{temp} に追加する。

【0096】

次に、分割線決定手段 10 は、関数分解可能性を判定する (S33)。関数分解の可能性の判定は、分割線で分割した場合の列複雑度 μ 及び集合 Z_{temp} の要素の数と LUT の入力数の最大値 k を比較することにより行われる。すなわち、関数分解可能な条件は、(数 17) で表される。従って、(数 17) の条件を満たすか否かで関数分解の可能性が判定される。

【0097】

【数 17】

$$\lceil \log_2 \mu \rceil < k \quad \text{かつ} \quad |Z_{temp}| \leq k$$

【0098】

上記ステップ S34 の判定の結果、関数分解が可能であれば、次に、変数 i が 1 であるか否かを判定する (S35)。 $i=1$ ならば、これ以上関数分解はできないので、LUT 生成手段 14 は関数 $f(Z_{temp})$ を LUT を生成し、LUT 記憶手段 16 に格納し、終了する (S36)。また、ステップ S35 において、分割線の高さ i が 1 よりも大きい場合には、分割線決定手段 10 は、集合 Z_a を集合 Z_{temp} に更新して (S37)、ステップ S31 に戻る。

【0099】

一方、ステップ S34 において、関数分解が不可能であると判定される場合、分割線決定手段 10 は、 Z_a の大きさ $|Z_a| = |H|$ かどうかを判定する (S38)。もし、 $|Z_a| = |H|$ ならば、これ以上分割線を下げても関数分解が可能となる可能性はない。従って、この場合には、分割線決定手段 10 は、LUT 論理回路が実現不可能である旨のメッセージを出力装置 4 に出力し (S39)、処理を終了する。

【0100】

ステップ S38 において、 $|Z_a| > |H|$ であれば、短絡除去手段 11 は、分割線の高さ以下の高さにある変数の集合 Z_b を $Z_t - Z_a$ とする (S40)。次に、短絡除去手段 11 は、分割線よりも高位の部分グラフに対して短絡除去を行う。次いで、BDD 幅計測手段 12 は、短絡除去がされた特性関数二分決定グラフについて、分割線における幅 μ_a を計測する。次に、中間変数算出手段 13 は、レイル数 u_a を (数 18) により計算する (S41)。

【0101】

【数 18】

$$u_a = \lceil \log_2 \mu_a \rceil$$

【0102】

次に、LUT生成手段14は、 (Z_a, Z_b) を変数の分割として、 $f(Z)=g(h(Z_a), Z_b)$ の形で関数分解したときの関数 $h(Z_a)$ のLUTを生成し、LUT記憶手段16に格納する(S42)。すなわち、上記S41の短絡処理が行われていない特性関数二分決定グラフについて、変数 Z_a に関する部分グラフを B_a 、 Z_b に関わる部分グラフを B_b とする。また、ステップS41において短絡除去がされた特性関数二分決定グラフについて、 $X_a (\subseteq Z_a)$ に関わる部分グラフを B_a' 、 Z_b に関わる部分グラフを B_b とする。LUT生成手段14は、部分グラフ B_a から、集合 Z_a に属するすべての入力変数 $x_i (\in Z_a)$ を制御入力とし、集合 Z_a に属するすべての出力変数 $y_j (\in Z_a)$ を出力するLUTを生成する。次いで、LUT生成手段14は、ステップS41において短絡除去がされた特性関数二分決定グラフについて、部分グラフ B_a' から直接接続される部分グラフ B_b 内の節点の各々に u_a ビットの符号 (h_1, \dots, h_{u_a}) を割り当てる。この符号 (h_1, \dots, h_{u_a}) を中間変数とする。そして、部分グラフ B_a' から、集合 Z_a に属するすべての入力変数 $X_a (\in Z_a)$ を制御入力とし、中間変数 (h_1, \dots, h_{u_a}) を出力するLUTを生成する。そして、これらのLUTをLUT記憶手段16に格納する。

【0103】

次に、BDD再構成手段15は、 (Z_a, Z_b) を変数の分割として、 $f(Z)=g(h(Z_a), Z_b)$ の形で関数分解したときの関数 $g(h, Z_b)$ についての特性関数二分決定グラフを生成する(S44)。すなわち、BDD再構成手段15は、ステップS41において短絡除去がされた特性関数二分決定グラフについて、 $X_a (\subseteq Z_a)$ に関わる部分グラフを、中間変数 (h_1, \dots, h_{u_a}) を制御入力とする二分木に置き換える。

【0104】

次に、中間変数の集合 H を、 u_a 個の中間変数 (h_1, \dots, h_{u_a}) に更新する。そして、集合 Z_t を集合 Z_b と集合 H との和集合 $Z_b \cup H$ とする(S44)。

【0105】

次に、 $|Z_t|$ が k 以下かどうかを判定する(S45)。 $|Z_t| \leq k$ ならば、関数 g のLUTを生成し、LUT記憶手段16に格納し(S46)、処理を終了する。

【0106】

ステップS44において、 $|Z_t| > k$ ならば、分割線の高さ i を $|Z_t| + 1$ 、 $B_{CF}^{current}$ を $B_{CF}(g)$ 、集合 Z_t を集合 Z_b と集合 H との和集合 $Z_b \cup H$ とした後(S47)、更に、集合 Z_a を集合 H 、関数 f を関数 g として(S48)、ステップS31に戻る。

【0107】

以上の処理により、論理関数の分解が行われ、LUTカスケード論理回路が構成される。

【0108】

以上のように、本実施例の論理回路合成装置1によれば、比較的少容量のメモリを用いてLUT論理回路の合成を実現できる。また、論理関数分解処理の演算処理速度を高速化することができる。

【0109】

また、現在まで有効な論理回路合成手段が知られていなかった、多出力論理関数の分解による途中出力を有するLUT論理回路の生成を、現実的なハードウェアを使用して短時間で実行することが可能となる。

【0110】

(5) 実験結果

最後に、本発明の効果を示すため、実際に幾つかのベンチマーク関数を用いてLUTカスケード論理回路の合成を行った結果を図16に示す。実験は上記実施例1のアルゴリズムをC言語で実装し、MCNC89ベンチマーク関数に適用した。図16は、LUTの入力数 k を10に設定した場合の結果である。

【0111】

図16において、「Name」は関数名、「In」は入力数、「Out」は出力数、「LUT」は生成されるLUTの総数、「Cas」はカスケードの個数を表し、「段数」はカスケードの段数を表す。実行環境は、IBM PC/AT互換機、Pentium (登録商標) 4 2.0GHz, メモリ 512MB

yte, OSはWindows (登録商標) 2000, cygwin上でgccを用いてコンパイルした。本アルゴリズムでは、出力のグループ分けを行うため、一つずつの出力をグループに加えて特性関数二分決定グラフを構成し、変数順序最適化を行う。kが大きくなると、LUTカスケードに対応する特性関数二分決定グラフが大きくなる。LUTカスケードで構成可能な限り、グループの出力を一つずつ増やしながらか変数順序最適化を行うため、大きな特性関数二分決定グラフの変数順序最適化を行う回数が増える。このため、kが大きくなると多くの実行時間が必要となる。実行時間の殆どは、特性関数二分決定グラフの変数順序最適化のための時間である。

【0112】

本発明の効果を確認するため、非特許文献15に記載の方法との比較を行った。非特許文献15の方法は、MTBDDに基づいており、多出力論理関数の出力を幾つかのグループに分割してMTBDDで表現し、LUTカスケードで実現する。MTBDDの幅が大きすぎるため分解不可能な場合は、OR分割を用いてカスケードを分割する。図16では、非特許文献15の方法については、一つのグループの出力数は8として、出力を分割している。

【0113】

非特許文献15の方法では、実現したLUTカスケードのLUTの個数や段数にかかわらず、分割するグループの出力数を固定しているため、カスケードの個数は多くなり、段数は小さくなる傾向にある。一方、本発明に係る方法では、LUTカスケードは可能な限り、一つのグループの出力の個数を増やすので、段数は多くなり、カスケードの個数は少なくなる。

【図面の簡単な説明】

【0114】

【図1】 特性関数二分決定グラフの一例を示す図である。

【図2】 出力変数の短絡除去を説明するための図である。

【図3】 多出力論理関数を2つの関数に関数分解した場合の論理回路の構成を表す図である。

【図4】 短絡除去を行った後の特性関数二分決定グラフの構成を表す図である。

【図5】 ADR2の特性関数二分決定グラフを表す図である。

【図6】 ADR2の特性関数二分決定グラフを分割したときの根節点側の部分グラフをLUTに変換する処理を説明するための図である。

【図7】 ADR2の特性関数二分決定グラフを短絡除去し、LUTに変換する処理を説明するための図である。

【図8】 ADR2の特性関数二分決定グラフを短絡除去し、LUTに変換する処理を説明するための図である。

【図9】 本発明の実施例1に係る論理回路合成装置及びその周辺装置の構成を表す図である。

【図10】 実施例1における論理回路合成方法の全体の流れを示すフローチャートである。

【図11】 変数順序の決定処理の流れを表すフローチャートである。

【図12】 節点テーブルのデータ構造を表す図である。

【図13】 (数15)の既約な特性関数二分決定グラフ及びその節点テーブルを表す図である。

【図14】 (数15)の特性関数二分決定グラフ及びその節点テーブルを表す図である。

【図15】 LUTカスケード論理回路の合成処理の流れを表すフローチャートである。

【図16】 本発明の論理回路合成方法によるLUTカスケード論理回路の合成についての実験結果を示す図である。

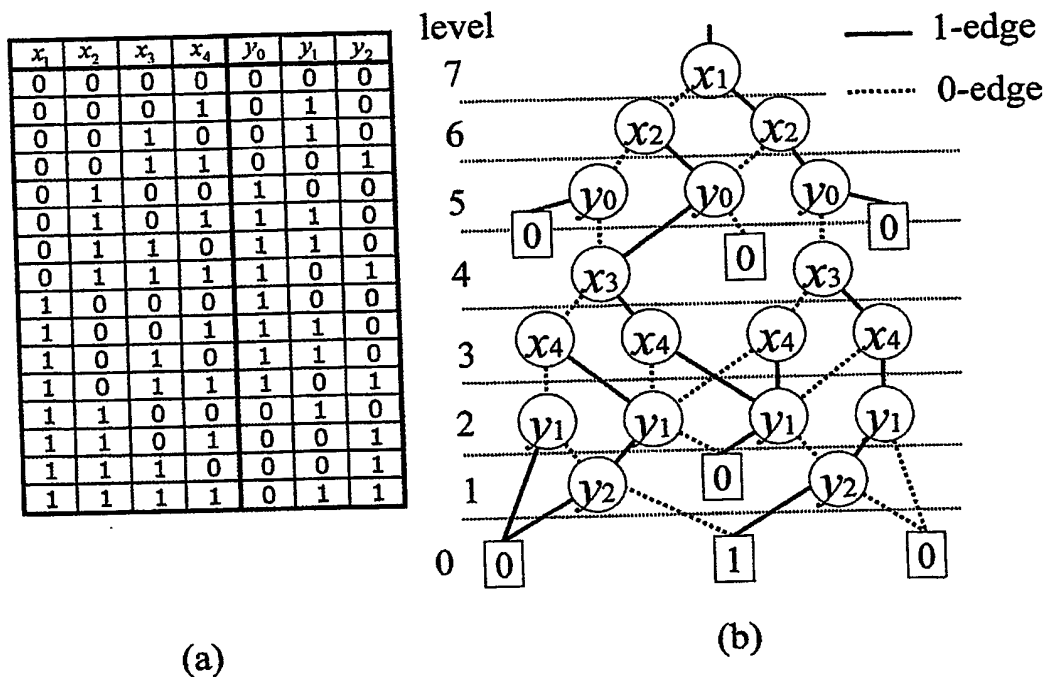
【符号の説明】

【0 1 1 5】

- 1 論理回路合成装置
- 2 論理仕様記憶手段
- 3 入力装置
- 4 出力装置
- 5 出力変数順序決定手段
- 6 全変数順序決定手段
- 7 BDD生成手段
- 8 節点テーブル記憶手段
- 9 変数順序最適化手段
- 1 0 分割線決定手段
- 1 1 短絡除去手段
- 1 2 BDD幅計測手段
- 1 3 中間変数算出手段
- 1 4 LUT生成手段
- 1 5 BDD再構成手段
- 1 6 LUT記憶手段

【書類名】図面

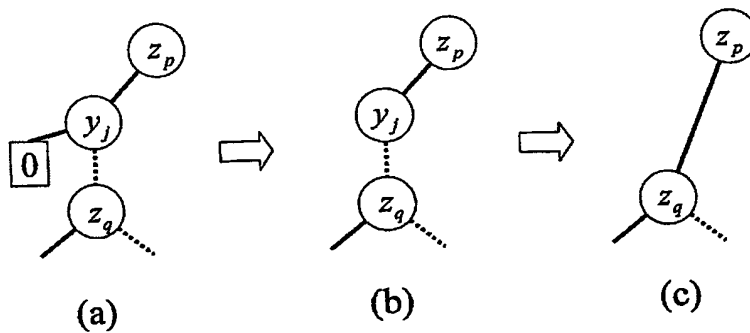
【図 1】



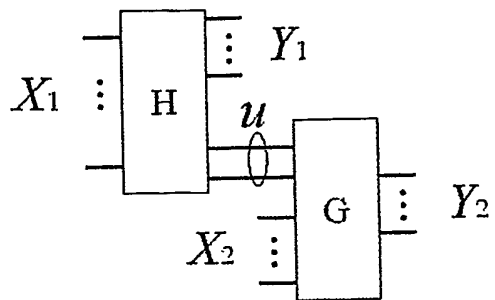
【図 2】

出力変数の短絡除去

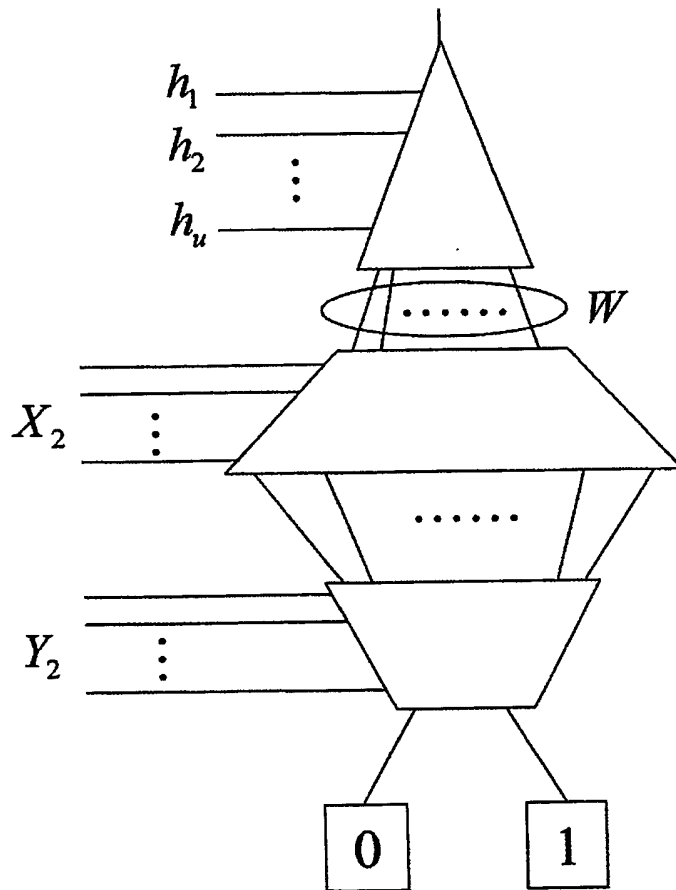
出力を表す節点 $y_j (\in Y)$ とその節点から出る枝のうち、定数0に接続する枝を取り除き、それ以外の枝が指す節点に、節点 y_j の親から直接枝を繋ぐ。



【図 3】



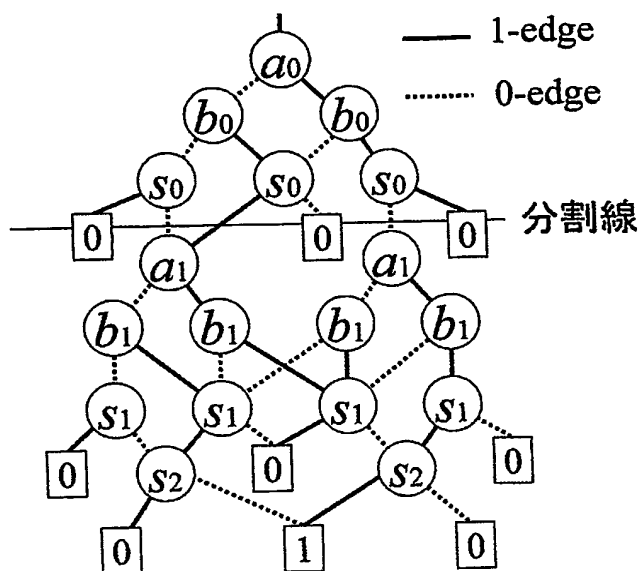
【図 4】



【図 5】

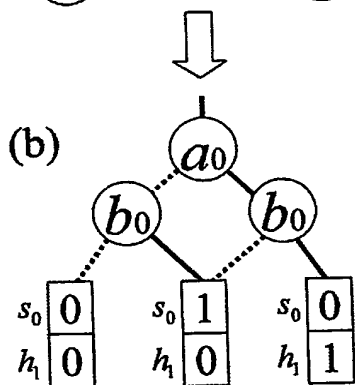
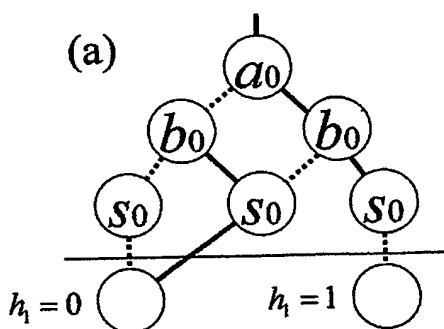
a_0	b_0	a_1	b_1	s_0	s_1	s_2
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	1	1	0
0	1	1	0	1	1	0
0	1	1	1	1	0	1
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	1	0	1
1	1	0	0	0	1	0
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	1

(a)



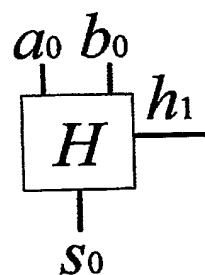
(b)

【図 6】

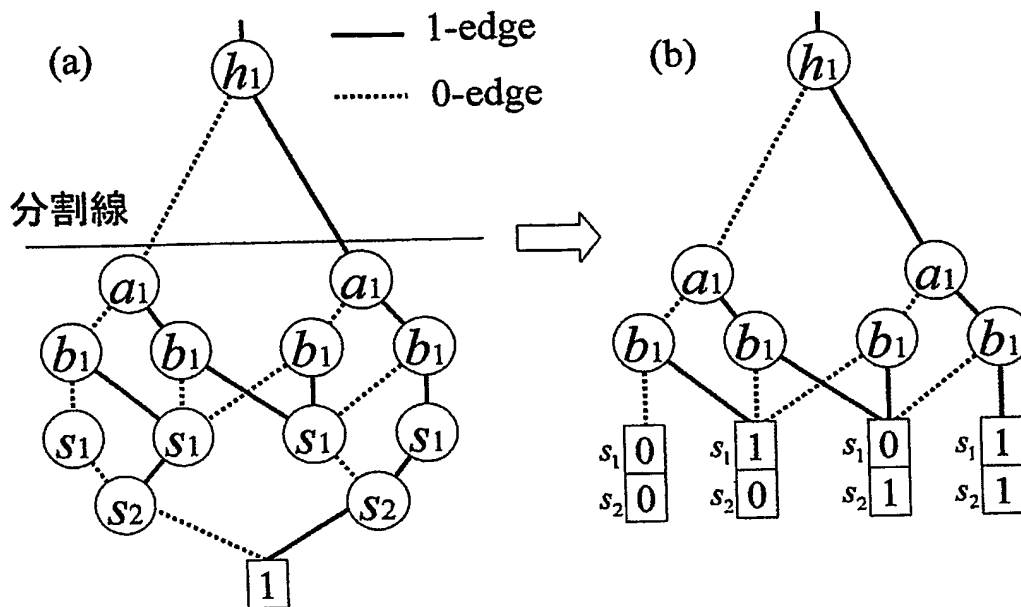


(c)

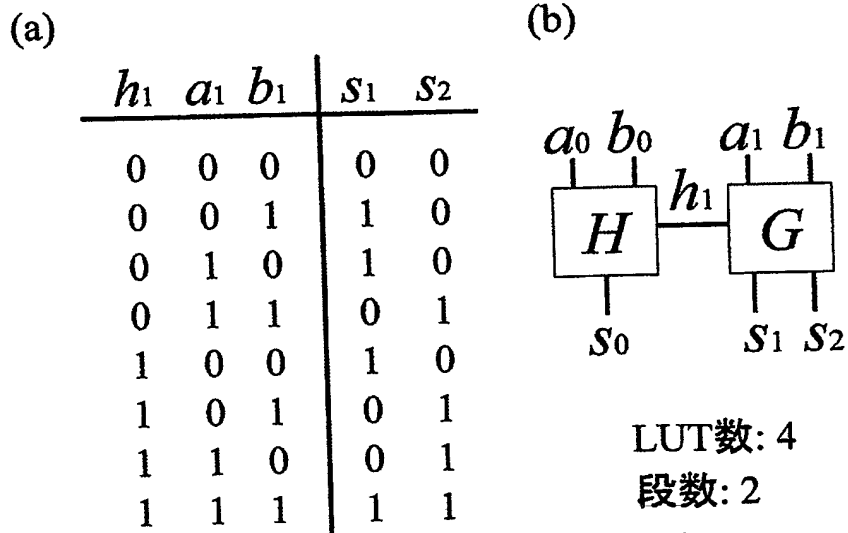
a_0	b_0	s_0	h_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



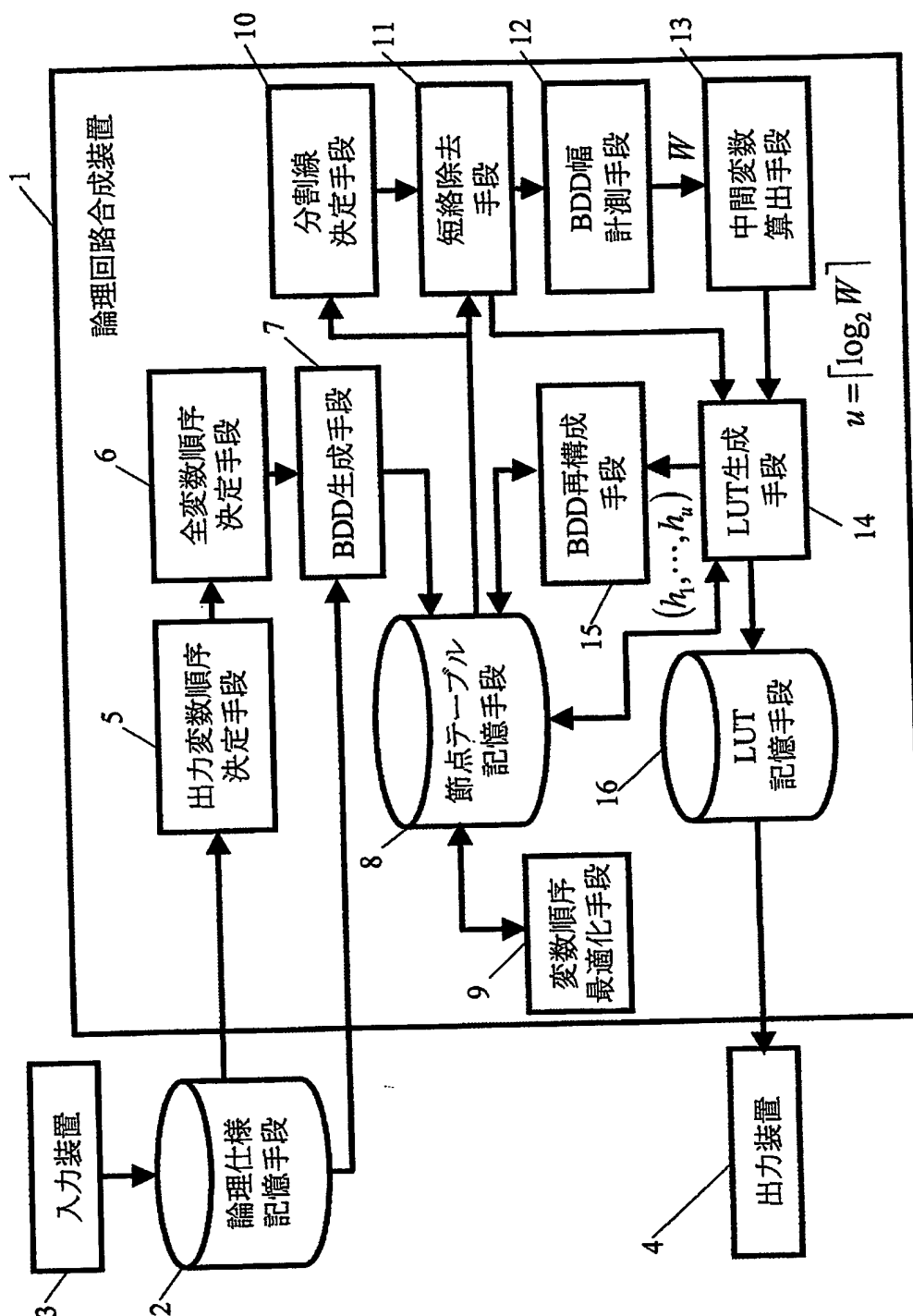
【図 7】



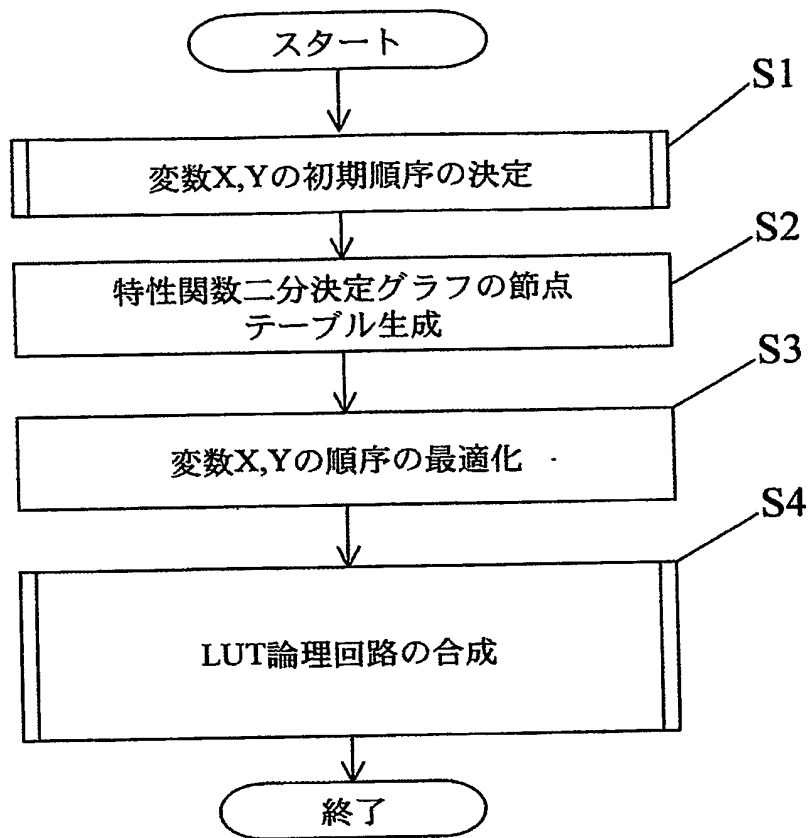
【図 8】



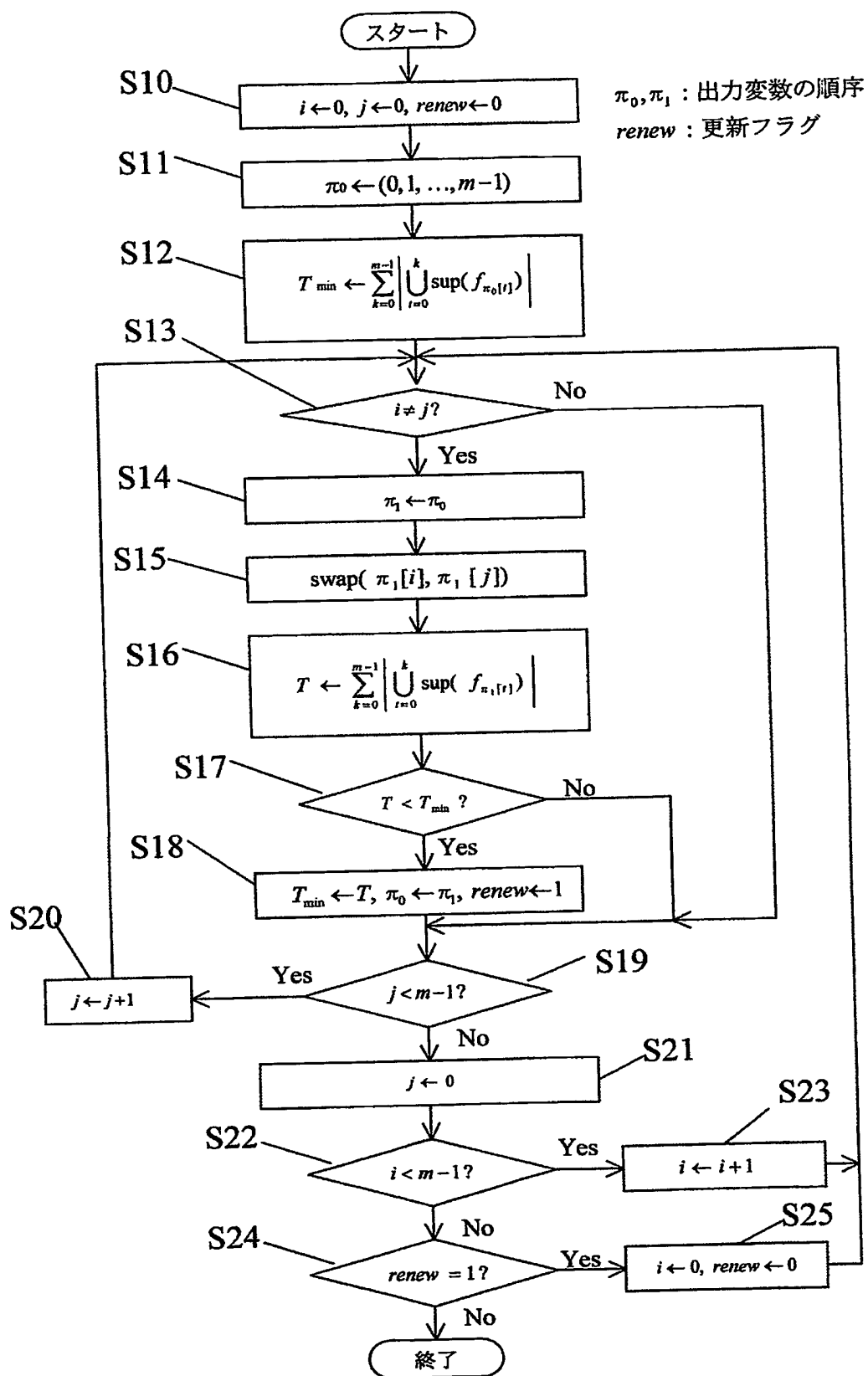
【圖 9】



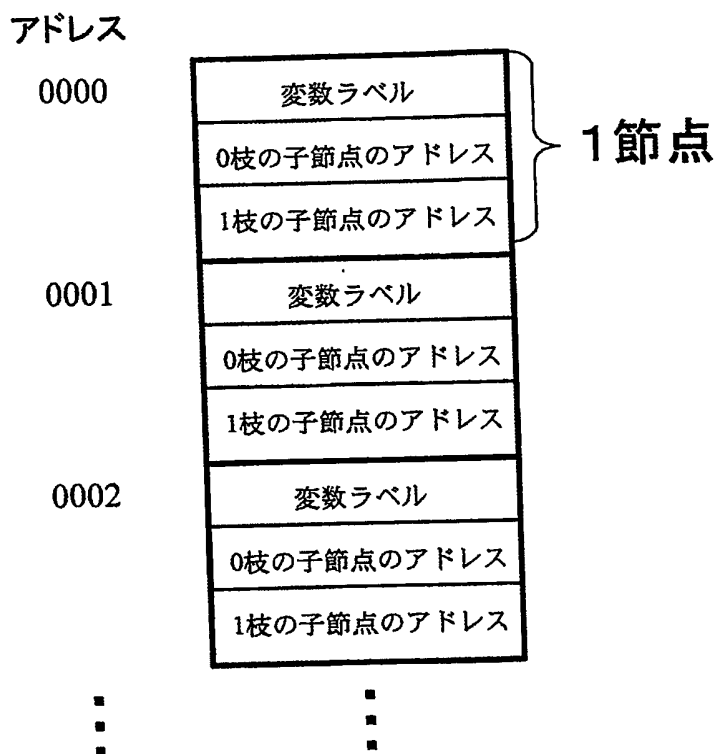
【図 10】



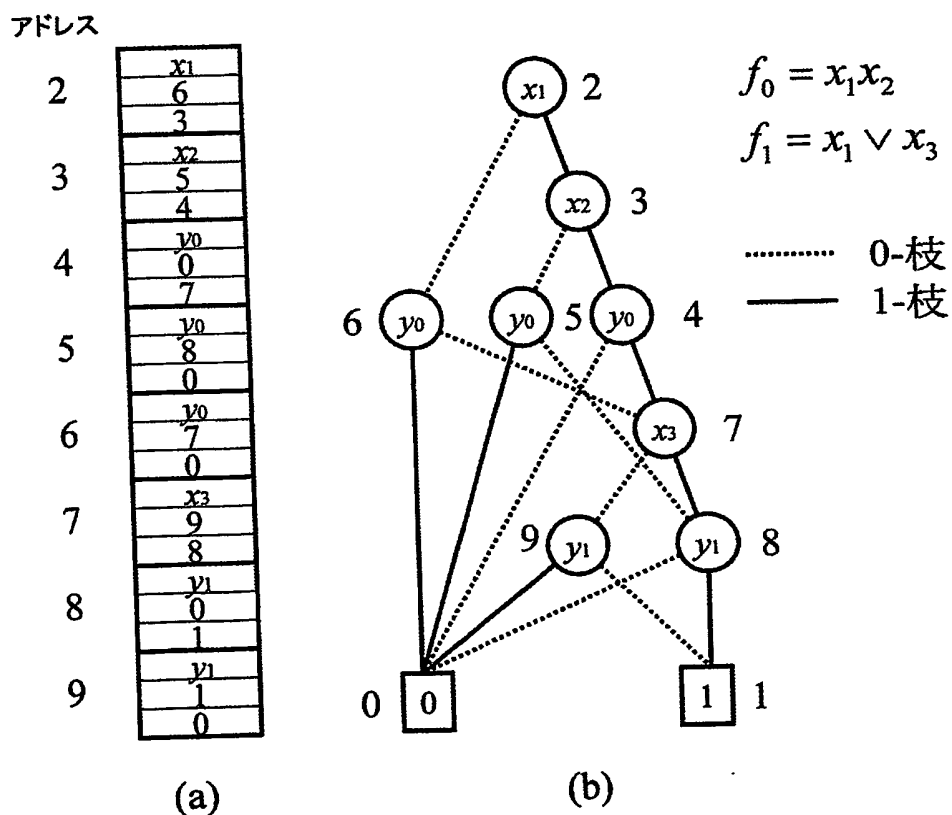
【図 11】



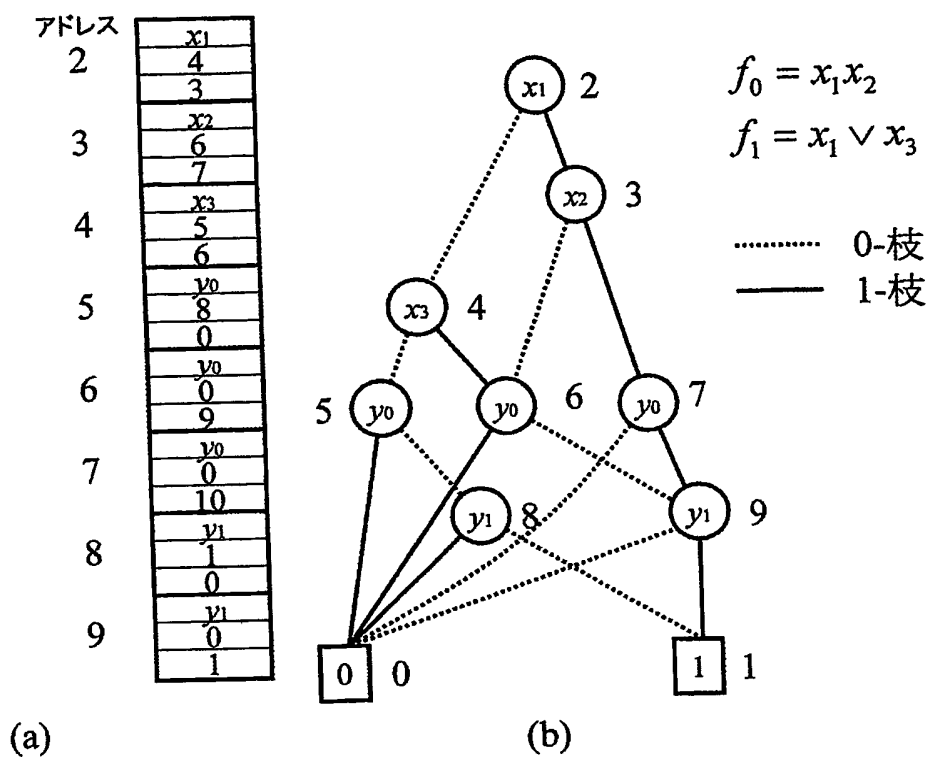
【図 12】



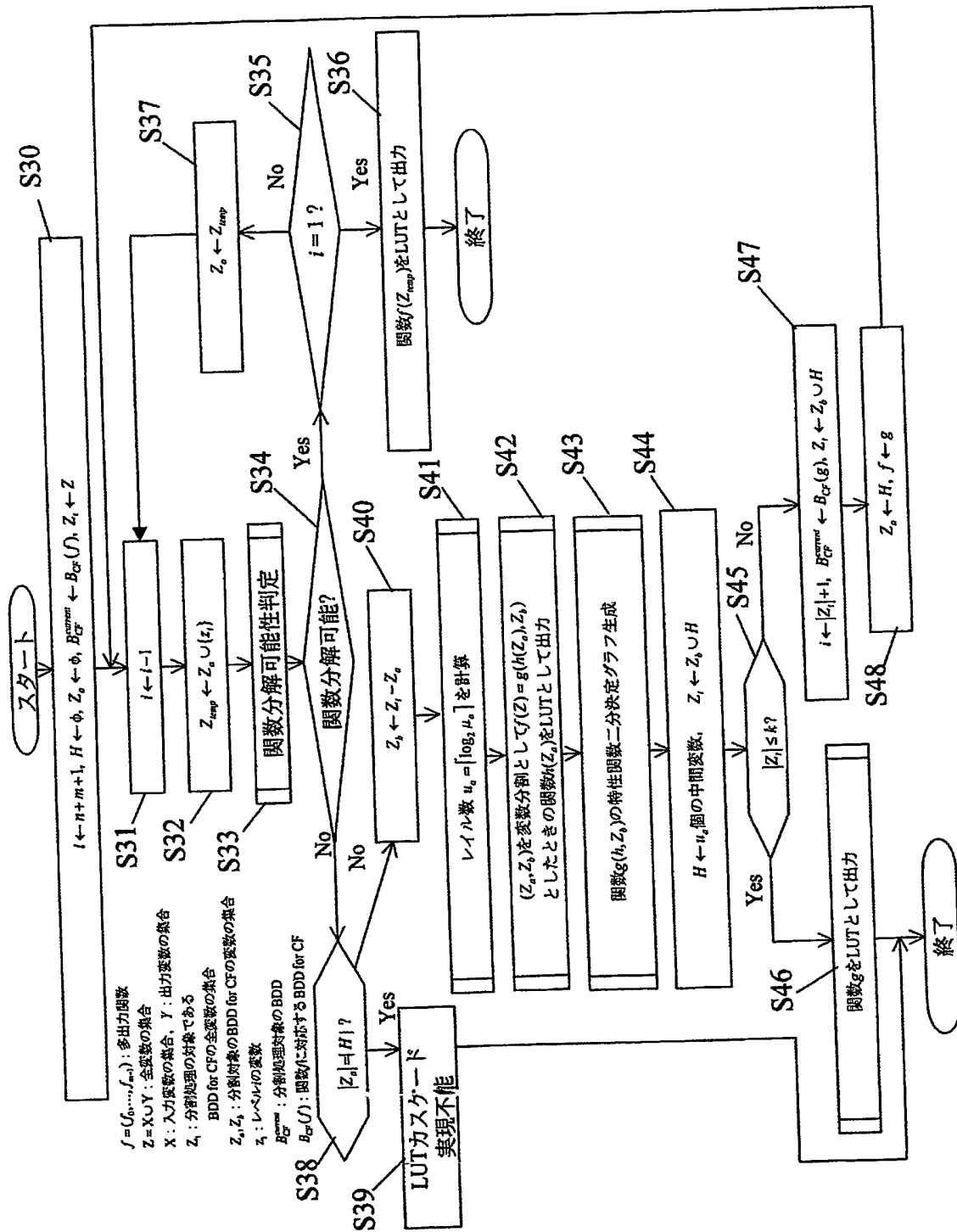
【図 13】



【図 14】



【図 15】



【図 16】

Name	In	Out	本発明の方法			Mishchenko [15]		
			LUT	段数	Cas	LUT	段数	Cas
C1908	33	25	320	11	5	3015	18	30
apex1	45	45	115	12	1	213	10	5
apex6	135	99	385	18	4	728	18	18
duke2	22	29	46	4	1	51	3	4
term1	34	10	50	8	1	37	6	2
ttt2	24	21	54	6	1	47	4	3

LUTの入力数を $k=10$ とした場合

【書類名】要約書

【要約】

【課題】多出力論理関数に対し、中間出力を有する L U T 論理回路が合成可能な論理回路合成装置を提供する。

【解決手段】多出力論理関数 $f(X)$ の特性関数 $\chi(X, Y)$ の特性関数二分決定グラフ節点テーブル記憶手段 8 と、L U T 記憶手段 16 と、特性関数二分決定グラフを所定の高さ lev の分割線で部分グラフ B_0, B_1 に分割し短絡除去処理を行う短絡除去手段 11 と、分割線における幅 W を計測する B D D 幅計測手段 12 と、幅 W に基づき中間変数の個数を算出する中間変数算出手段 13 と、部分グラフ B_0 につき L U T を生成する L U T 生成手段 14 と、中間変数の個数 u と等しい制御入力数を有する二分木を生成し、部分グラフ B_0 を二分木で置き換え、特性関数二分決定グラフを再構成する B D D 再構成手段 15 とを備えた構成とした。

【選択図】

図 9

認定・付加情報

特許出願の番号	特願2003-389264
受付番号	50301910135
書類名	特許願
担当官	末武 実 1912
作成日	平成15年11月20日

<認定情報・付加情報>

【提出日】

平成15年11月19日

特願 2003-389264

出 願 人 履 歴 情 報

識別番号

[802000031]

1. 変更年月日

2002年 4月19日

[変更理由]

新規登録

住 所

福岡県北九州市若松区ひびきの2番1号

氏 名

財団法人北九州産業学術推進機構

Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/JP04/017263

International filing date: 19 November 2004 (19.11.2004)

Document type: Certified copy of priority document

Document details: Country/Office: JP
Number: 2003-389264
Filing date: 19 November 2003 (19.11.2003)

Date of receipt at the International Bureau: 27 January 2005 (27.01.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)



World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse